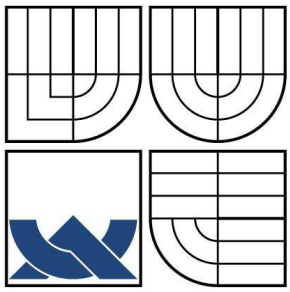
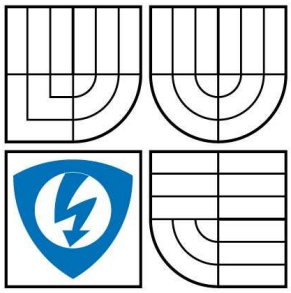


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

TURBOKÓDY A JEJICH POUŽITÍ VE SDĚLOVACÍCH SYSTÉMECH

TURBOCODES AND THEIR APPLICATION IN TELECOMMUNICATION SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. TRČKA TOMÁŠ
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. KAREL NĚMEC, CS.c
SUPERVISOR

ANOTACE

Tato diplomová práce se zabývá problematikou Turbokódů. Tyto kódy patří do skupiny samoopravných kódů, někdy též nazývaných jako dopředné korekční kódy nebo kanálové kódy. Práci lze tématicky rozdělit na dvě základní části. První část popisuje blokové schéma kodéru turbokódu, dekodéru turbokódu, součástí je i ukázka dvou nejpoužívanějších algoritmů pro iterativní dekódování turbokódů (SOVA a MAP). Závěr této části tvoří seznam nejznámějších turbokódů používaných ve sdělovacích systémech současnosti.

Druhá část se věnuje prezentaci výsledků simulace zabezpečovacích schopností turbokódů v programu MATLAB/SIMULINK. Přenosová cesta byla simulována AWGN kanálem při použití modulace BPSK. Bude zde ukázáno, že existuje mnoho různých parametrů, které významným způsobem ovlivňují výkonnost turbokódů. Některé z těchto parametrů jsou například: počet použitých iterací při dekódování, délka vstupního bloku dat, generující polynomy a délka kódového ohraničení RSC kodérů, vhodně navržený blok prokládání, použitý dekódovací algoritmus, atd.

Klíčová slova: kodér turbokódu, dekodér turbokódu, iterativní dekódování, SOVA, MAP, seznam nejznámějších turbokódů, výsledky simulace v programu MATLAB/SIMULINK.

ABSTRACT

This Diploma thesis deals with Turbo code problems. The Turbo codes belong to the group of error correction codes, sometimes referred to as forward error correcting (FEC) codes or channel codes. This thesis can be thematically divided into two basic parts. The first part describes turbo code encoder and decoder block diagram with the illustration of two most frequently used iterative decoding algorithms (SOVA and MAP). The end of this part contains best known turbo codes, which are used in present communication systems.

The second part pursues simulation results for the turbo codes using Binary Phase Shift Keying (BPSK) over Additive White Gaussian Noise (AWGN) channels. These simulations were created in the MATLAB/SIMULINK computer program. It will be shown here, that there exist many different parameters, greatly affecting turbo codes performance. Some of these parameters are: number of decoding iterations used, the input data frame length, generating polynoms and RSC encoders constraint lengths, properly designed interleaving block, decoding algorithm used, etc.

Keywords: turbo code encoder, turbo code decoder, iterative decoding, SOVA, MAP, list of best known turbo codes, simulation resuslts in MATLAB/SIMULINK program.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma turbokódy a jejich použití ve sdělovacích systémech jsem vypracoval samostatně pod vedením vedoucího semestrálního projektu a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedeného semestrálního projektu dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

Obsah

OBSAH **7**
ÚVOD **11**
1 KÓDOVÁNÍ TURBOKÓDŮ **13**
1.1 OBECNÉ BLOKOVÉ SCHÉMA TURBOKODÉRU 13
1.2 NEREKURZIVNÍ KONVOLUČNÍ KODÉRY (NRC) 14
1.3 REKURZIVNÍ SYSTEMATICKÉ KONVOLUČNÍ KODÉRY (RSC) 14
1.4 UKONČOVACÍ BITY 15
1.5 ZABEZPEČOVACÍ SCHOPNOSTI KÓDŮ 16
1.6 PODROBNĚJŠÍ ROZBOR VLASTNOSTÍ RSC KODÉRŮ 16
1.6.1 Snadná realizace 16
1.6.2 Systematičnost 16
1.6.3 Rekurzivní povaha 17
1.6.4 Nekonečná impulsní odezva IIR (infinite impulse response) 17
1.7 BLOK PROKLÁDÁNÍ 18
1.8 DĚROVÁNÍ 20
2 DEKÓDOVÁNÍ TURBOKÓDŮ **21**
2.1 DEKÓDOVÁNÍ KONVOLUČNÍCH KÓDŮ – VITERBI ALGORITMUS 21
2.2 DEKÓDOVACÍ NÁROČNOST PRO KONVOLUČNÍ KÓDY 25
2.3 ITERATIVNÍ DEKÓDOVÁNÍ TURBOKÓDŮ 25
2.4 SOFT - OUTPUT VITERBI (SOVA) ALGORITMUS 26
2.5 MAXIMUM A-POSTERIORI (MAP) ALGORITMUS 30
2.5.1 Průchod v dopředném směru 30
2.5.2 Průchod ve zpětném směru 31
2.5.3 Výpočet pravděpodobnosti jednotlivých přechodů 31
2.5.4 Výpočet měkkého výstupu pro dekódované bity 31
3 VYUŽITÍ TURBOKÓDŮ VE SDĚLOVACÍCH SYSTÉMECH **33**
3.1 NEJZNÁMĚJŠÍ APLIKACE TURBOKÓDŮ VE SDĚLOVACÍCH SYSTÉMECH 33
3.2 TURBOKÓDY PRO SYSTÉMY 3G 33
3.2.1 UMTS turbokód 33
3.2.2 cdma2000 turbokód 34
3.2.3 Dekódování UMTS a cdma2000 turbokódů 34
3.3 TURBOKÓDY POUŽITÉ V DVB (DIGITAL VIDEO BROADCASTING) 35
3.3.1 DVB–RCS (Return Channel over satellite) turbokód 35
3.3.2 DVB–RCT (Return Channel over Terrestrial) turbokód 36
3.4 CCSDS TURBOKÓD 36
3.4 DALŠÍ TURBOKÓDY POUŽÍVANÉ VE SDĚLOVACÍCH SYSTÉMECH SOUČASNOSTI 37
3.4.1 INMARSAT (M4) 37
3.4.2 EUTELSAT (Skyplex) 37
3.4.3 IEEE 802.16 (WiMAX) 37
4 PARAMETRY A POPIS SIMULACE **39**
4.1 ZÁKLADNÍ POPIS MODELU POUŽITÉHO PŘI SIMULACI 39
4.2 PROBLEMATIKA UKONČOVACÍCH BITŮ 40

4.3 VNITŘNÍ STRUKTURA TURBOKODÉRU 41
4.4 VNITŘNÍ STRUKTURA TURBODEKODÉRU 42
4.5 POPIS DŮLEŽITÝCH ČÁSTÍ SKRIPTU V PROGRAMU MATLAB 43
4.5.1 Inicializace proměnných 43
4.5.2 Nastavení bloku konvolučního kodéru 43
4.5.3 Nastavení bloku AWGN kanálu 44
4.5.4 Realizace iterativního dekódování 45
4.6 ROZŠÍŘENÍ ZÁKLADNÍHO MODELU O DĚROVÁNÍ 46
4.7 MODEL ZABEZPEČOVACÍHO PROCESU U KONVOLUČNÍCH KÓDŮ 47

5 PREZENTACE VÝSLEDKŮ SIMULACE 48

5.1 VLIV POČTU DEKÓDOVACÍCH KROKŮ 48
5.2 VLIV DĚROVÁNÍ NA VÝKONNOST TURBOKÓDU 49
5.3 VLIV ZMĚNY DÉLKY KÓDOVÉHO OHRANIČENÍ RSC KODÉRŮ 51
5.4 VLIV SPRÁVNÉ VOLBY GENERUJÍCÍCH POLYNOMŮ 53
5.5 VLIV VELIKOSTI VSTUPNÍHO BLOKU DAT 54
5.6 VLIV POUŽITÉHO PROKLADAČE NA VÝKONNOST TURBOKÓDU 55
5.7 VLIV POUŽITÉHO DEKÓDOVACÍHO ALGORITMU 56
5.8 POPIS PARAMETRU W A JEHO VLIV NA VÝKONNOST TURBOKÓDU 57

6 ZÁVĚR 59

7 POUŽITÁ LITERATURA 61

Seznam obrázků a tabulek

OBR. 1.1: OBECNÉ BLOKOVÉ SCHÉMA KODÉRU TURBOKÓDU. 13
OBR. 1.2: ZAPOJENÍ NRC KODÉRU S $R = 1/2$ A $K = 4$. 14
OBR. 1.3: ZAPOJENÍ RSC KODÉRU ODVOZENÉ Z ODPOVÍDAJÍCÍHO NRC KODÉRU. 15
OBR. 1.4: ZAPOJENÍ RSC KODÉRU UMOŽŇUJÍCÍ VYNULOVÁNÍ PAMĚŤOVÝCH BUNĚK. 15
OBR. 1.5: ZAPOJENÍ TURBOKODÉRU S $R = 1/3$ 18
OBR. 1.6: UKÁZKA MOŽNÝCH ZPŮSOBŮ NÁVRHU PROKLADAČE. 20
OBR. 1.7: DĚROVACÍ VZORY PRO INFORMAČNÍ RYCHLOSTI $R = 1/2, R = 3/4$ 20
OBR. 2.1: ZAPOJENÍ RSC KODÉRU. 21
OBR. 2.2: JEDEN STAVOVĚ-ČASOVÝ KROK MŘÍŽOVÉHO GRAF. 22
OBR. 2.4: OBECNÉ BLOKOVÉ SCHÉMA DEKODÉRU TURBOKÓDU. 26
OBR. 2.6: UKÁZKA VÝPOČTU STAVOVÝCH PRAVDĚPODOBNOSTÍ $\alpha_k(1)$ A $\beta_k(1)$. 32
OBR. 3.1: UKÁZKA ZAPOJENÍ (A) UMTS TURBOKODÉRU, (B) CDMA2000 TURBOKODÉRU. 34
OBR. 3.2: ARCHITEKTURA PRO SPOLEČNÉ DEKÓDOVÁNÍ UMTS A CDMA2000 TURBOKÓDU. 35
OBR. 3.3: SCHÉMA KODÉRU PRO DVB-RCS TURBOKÓD. 35
OBR. 3.4: SCHÉMA KODÉRU PRO CCSDS TURBOKÓD. 36
OBR. 3.5: UKÁZKA DALŠÍCH, ČASTO POUŽÍVANÝCH TURBOKÓDŮ Z TABULKY 3.1. 38
OBR. 4.1: ZÁKLADNÍ MODEL POUŽITÝ PŘI SIMULACI (BEZ DĚROVÁNÍ). 39
OBR. 4.2: UKONČOVACÍ BITY SYSTEMATICKÉ POSLOUPNOSTI JSOU ODVOZENY V KODÉRU RSC1. 40
OBR. 4.3: UKONČOVACÍ BITY SYSTEMATICKÉ POSLOUPNOSTI JSOU ODVOZENY V KODÉRU RSC2. 40
OBR. 4.4: SCHÉMA ZAPOJENÍ TURBOKODÉRU POUŽITÉHO PŘI SIMULACI. 41
OBR. 4.5: SCHÉMA ZAPOJENÍ TURBODEKODÉRU POUŽITÉ PŘI SIMULACI. 42
OBR. 4.6: ČÁST SKRIPTU – INICIALIZACE PROMĚNNÝCH. 43
OBR. 4.7: UKÁZKA ZAPOJENÍ RSC KODÉRU. 44
OBR. 4.8: ILUSTRAČNÍ OBRÁZEK BLOKU TURBOKODÉRU. 44
OBR. 4.9: ČÁST SKRIPTU – REALIZACE ITERATIVNÍHO DEKÓDOVÁNÍ. 45
OBR. 4.10: ROZŠÍŘENÍ ZÁKLADNÍHO MODELU O DĚROVÁNÍ. 46
OBR. 4.11: MODEL PRO SIMULACI ZABEZPEČOVACÍHO PROCESU U KONVOLUČNÍCH KÓDŮ. 47
OBR. 5.1: VLIV POČTU DEKÓDOVACÍCH KROKŮ. 49
OBR. 5.2: VLIV DĚROVÁNÍ NA VÝKONNOST TURBOKÓDŮ. 50
OBR. 5.3: VLIV DĚROVÁNÍ NA VÝKONNOST TURBOKÓDŮ - SROVNÁNÍ. 51
OBR. 5.4: VLIV DÉLKY KÓDOVÉHO OHRANIČENÍ. 52
OBR. 5.5: VLIV VOLBY GENERUJÍCÍCH POLYNOMŮ. 54
OBR. 5.6: VLIV DÉLKY VSTUPNÍHO BLOKU DAT. 55
OBR. 5.7: VLIV PROKLADAČE NA VÝKONNOST TURBOKÓDU. 56
OBR. 5.8: VLIV POUŽITÉHO DEKÓDOVACÍHO ALGORITMU – PŘEVZATO Z LITERATURY [17]. 57
OBR. 5.9: VLIV PARAMETRU W NA VÝKONNOST TURBOKÓDU. 58

TAB. 2.1: TABULKA POPISUJÍCÍ JEDNOTLIVÉ STAVY KONVOLUČNÍHO KODÉRU. 21
TAB. 2.2: TABULKA HODNOT d_A PRO VŠECH 13 ÚSEKŮ MŘÍŽOVÉHO GRAFU. 24
TAB. 2.3: STAVY PŘI ZPĚTNÉM TRASOVÁNÍ. 24
TAB. 2.4: DEKÓDOVANÁ ZAPAMATOVANÉ VSTUPNÍ POSLOUPNOST. 25
TAB. 2.5: TABULKA HODNOT AKUMULOVANÉ METRIKY PRO JEDNOTLIVÉ UZLY. 30
TAB. 2.6: TABULKA JEDNOTLIVÝCH VÝPOČTŮ U SOVA ALGORITMU. 30
TAB. 3.1: NEJZNÁMĚJŠÍ APLIKACE TURBOKÓDŮ VE SDĚLOVACÍCH SYSTÉMECH. 33
TAB. 5.1: POČET CHYB V JEDNOTLIVÝCH DEKÓDOVACÍCH KROCÍCH. 48
TAB. 5.2: NEJVÝKONNĚJŠÍ KONVOLUČNÍ KÓDY S KÓDOVOU RYCHLOSTÍ $R = 1/2$. 53

Seznam použitých zkratek

ATM	Asynchronous Transfer Mode
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BCH	Bose-Chaudhuri-Hocquenghem (Code)
BPSK	Binary Phase Shift Keying
CCSDS	Consultative Commitee for Space Data Systems
CDMA	Code Division Multiple Access
CRSC	Circular Recursive Systematic Convolutional (Code)
DVB	Digital Video Broadcasting
DVB-RCS	Digital Video Broadcasting - Return Channel over Satellite
DVB-RCT	Digital Video Broadcasting - Return Channel over Terrestrial
FEC	Forward Error Correction
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
IP	Internet Protocol
MAP	Maximum A Posteriori (Algorithm)
Log-MAP	Logarithmic-MAP (Algorithm)
Max-Log-MAP	Maximum-Logarithmic-MAP (Algorithm)
MPEG-2	Motion Picture Expert Group - Standard 2
NRC	Non-Recursive Convolutional (Code)
QoS	Quality of Service
RS	Reed-Solomon (Code)
RSC	Recursive Systematic Convolutional (Code)
SNR	Signal to Noise Ratio
SOVA	Soft Output Viterbi Algorithm
UMTS	Universal Mobile Telecommunications System
WiFi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
3G	Third Generation Mobile Systems
3GPP	Third Generation Partnership Project

Úvod

Kanálové kódování se často používá v digitálních komunikačních systémech z důvodu ochrany digitálních informací před šumem a rušením a tím přispívá k redukci počtu bitových chyb. Při kanálovém kódování dochází ke „*kontrolovanému*“ přidávání určitého počtu redundantních informací do přenášeného datového toku. Tyto nadbytečné informace potom umožňují detekci a korekci bitových chyb na straně přijímače a tím zajišťují větší spolehlivost datového přenosu. Cenou za používání kanálového kódování k zabezpečení přenášených informací je buď snižování přenosové rychlosti, nebo odpovídající nárůst potřebné šířky pásma. Mezi dva základní typy kanálových kódů patří blokové kódy a konvoluční kódy.

Blokové kódy na svém vstupu přijímají blok dat o délce *k* informačních bitů a na výstupu produkují blok o délce *n* zakódovaných bitů. Ke vstupnímu bloku dat je tedy podle předem daných pravidel přidáno $n-k$ nadbytečných bitů. Do této skupiny kódů patří například Hammingovy kódy, BCH kódy a RS kódy.

Konvoluční kódy jsou jedny z nejvíce rozšířených kanálových kódů používaných v komunikačních systémech. V případě konvolučních kódů se využívá jiného způsobu zabezpečení vstupních bitů než u kódů blokových. Základním prvkem konvolučního kodéru je posuvný registr složený z několika paměťových buněk, které v sobě uchovávají předchozí hodnoty vstupního bitového toku. Jednotlivé výstupy tohoto kodéru jsou tvořeny lineární kombinací současných a předchozích vstupních hodnot. K dekódování konvolučních kódů se používá Viterbi algoritmus.

Shannon ve svém teorému kanálového kódování odvodil, že frekvence výskytu chyb v datech přenášených pásmově omezeným kanálem za přítomnosti šumu, může být vhodným kódováním redukována na libovolně malou hodnotu, pokud je rychlost přenosu informace menší než kapacita přenosového kanálu [1]. Pro AWGN (Additive White Gaussian Noise) kanál je přenosová kapacita C určena *Shannon-Hartleyovým* vztahem:

$$C = B\log_2\left(1+\frac{S}{N}\right)\ [\text{bit/s}], \tag{0.1}$$

kde B udává šířku pásma přenosového kanálu, S je střední hodnota výkonu užitečného signálu na vstupu přijímače a symbol N označuje střední hodnotu výkonu šumu.

Shannon rozvinul jeho teorii již v roce 1940, avšak ani několik desetiletí poté nebyly navržené kanálové kódy schopny přiblížit se významným způsobem tomuto teoretickému limitu. Ještě v roce 1990 byla mezera mezi teoretickou hranicí a praktickými implementacemi asi 3 dB. Z toho důvodu byly hledány nové kódy, které by byly schopny více se přiblížit Shannonovu limitu.

Jedna z myšlenek byla využít kombinace jednoduchých kódů v paralelním zřetězení tak, aby každý z těchto kódů mohl být dekódován odděleně v méně složitém dekodéru. Každý z takových dekodérů by navíc mohl těžit ze vzájemné výměny informací mezi jednotlivými dekodéry.

Claude Berrou, Alain Glavieux a Punya Thitimajshima publikovali v roce 1993 článek [6], ve kterém byly poprvé představeny turbokódy. Autoři v tomto článku ukázali, že turbokódy umožňují dosáhnout nízké chybovosti (BER-Bit Error Rate) i při hodnotách SNR, které leží velmi blízko Shannonovu limitu (v literatuře [16] je uvedena vzdálenost pouhých 0,7 decibelů od Shannonova limitu pro $BER = 10^{-5}$). Před rokem 1993 převládala

myšlenka, že se k této teoretické hranici můžeme přiblížit pouze použitím kódů s extrémně dlouho délkou kódového slova, jejichž případné dekódování by bylo ovšem velmi složité. Berrou a jeho skupina ve svém článku předvedli, jakým způsobem je možné přiblížit se Shannonovu limitu i s realizovatelnou dekódovací náročností. Myšlenka turbokódů vychází z představy paralelně zřetězených kódů (vzájemně oddělených blokem prokládání) a iterativního způsobu dekódování.

Cílem této diplomové práce je seznámení se s problematikou návrhu a dekódování turbokódů, nalezení jejich využití ve sdělovacích systémech současnosti a následná simulace procesu kódování a dekódovaní turbokódů na osobním počítači. Práci jsem strukturoval do šesti základních kapitol.

První kapitola popisuje základní blokovou strukturu turbokodéru. Jsou v ní postupně rozebrány jeho jednotlivé části včetně jejich vlivu na celkové vlastnosti a výkonnost celého turbokódu. Největší pozornost je v této kapitole věnována detailnějšímu popisu vlastností RSC kodérů a bloku prokládání.

Začátek druhé kapitoly zahrnuje ukázku dekódování konvolučních kódů pomocí Viterbi algoritmu. Následuje objasnění principu iterativního dekódování turbokódů a poté jsou postupně vysvětleny dva nejčastěji používané dekódovací algoritmy SOVA a MAP. U Viterbi a SOVA algoritmu jsem jejich funkci předvedl na ilustrativním příkladu. V případě MAP algoritmu jsem, z důvodu jeho výpočetní náročnosti, uvedl pouze obecný postup při dekódování.

Ve třetí kapitole je ukázka využití některých turbokódů v současných sdělovacích systémech. Turbokódy, využívané v nejznámějších sdělovacích systémech jako jsou UMTS, cdma2000, DVB-RCS, DVB-RCT a CCSDS, jsou rozebrány podrobněji.

Čtvrtá kapitola se věnuje detailnějšímu popisu jednotlivých modelů a jejich bloků, použitých k simulaci korekčních schopností turbokódů v programu MATLAB/SIMULINK. Dále jsou v této kapitole podrobněji rozebrány části skriptu důležité pro správné nastavení parametrů simulace a jejich případné modifikace.

Pátá kapitola je prezentací výsledků jednotlivých simulací. Jsou zde postupně uváděny parametry turbokódů, které je potřeba zohlednit při jejich návrhu, jelikož mají největší vliv na celkovou výkonnost daného turbokódu. Mezi tyto parametry patří například počet dekódovacích kroků turbodekodéru, délka bloku vstupních dat, délka kódového ohraničení v jednotlivých RSC kodérech, použití děrování a další.

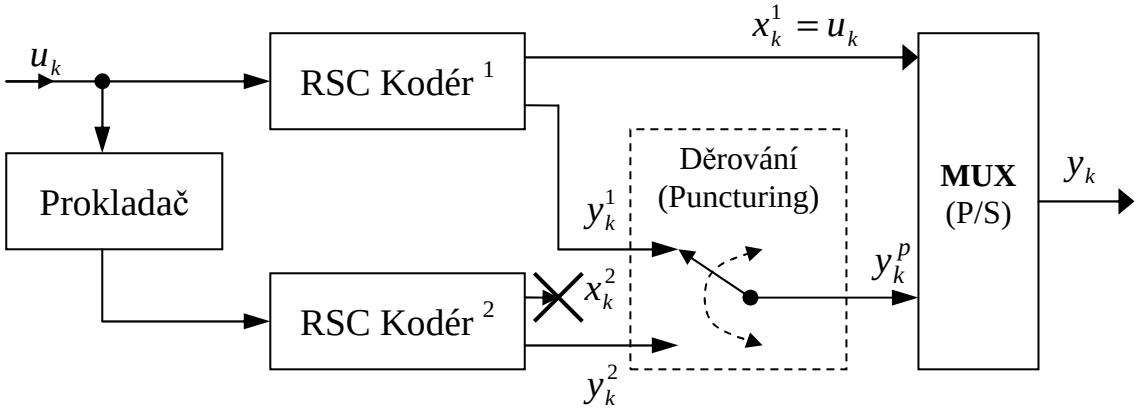
Poslední kapitola tvoří závěr této diplomové práce. Jsou zde zhodnoceny a shrnuty důležité poznatky z jednotlivých kapitol, převážně z kapitoly páté.

1 Kódování turbokódů

Tato kapitola se opírá o literaturu [3, 8, 10, 11, 13].

1.1 Obecné blokové schéma turbokodéru

Ve většině případů se turbokodér skládá ze dvou (případně více) paralelně zřetězených konvolučních kódů. V praxi se nejčastěji používají dva identické rekurzivní systematické konvoluční kodéry (RSC), které jsou od sebe vzájemně odděleny prokladačem. Obrázek 1.1 ukazuje obecné blokové schéma turbokodéru.



Obr. 1.1: Obecné blokové schéma kodéru turbokódu.

Proces zabezpečení začíná tím, že je na vstup turbokodéru přivedena vstupní nezabezpečená posloupnost $u = \{u_1, u_2, u_3, ..., u_N\}$. Informační rychlost RSC kodérů je typicky $R = 1/2$. Vstupní data jsou přivedena do prvního RSC kodéru přímo a do druhého RSC kodéru přes prokladač. Ten převede přijatou vstupní posloupnost na novou posloupnost, která bude mít pseudonáhodný tvar. Každý z RSC kodéru na svém výstupu poskytuje systematickou a zabezpečovací (paritní) posloupnost. Systematická část z druhého RSC kodéru není dále přenášena, protože je možné ji v dekodéru následně obnovit ze systematické části prvního RSC dekodéru.

Pokud chceme zlepšit výslednou informační rychlost turbokodéru, je možné použít blok děrování (puncturing). Tento blok podle daných pravidel odstraní některé bity z paritních posloupností y_k^1 a y_k^2. Takto vzniklá zabezpečující posloupnost y_k^p je společně se systematickou částí z prvního RSC kodéru přivedena na paralelně/sériový převodník (P/S). Při informační rychlosti $R = 1/2$ obou RSC kodérů a bez použití bloku děrování je výsledná informační rychlost turbokodéru rovna $R_v = 1/3$. Pro každý vstupní datový bit u_k dostaneme na výstupu turbokodéru trojici bitů $y_k = \{x_k^1, y_k^1, y_k^2\}$. Výstupní zabezpečená posloupnost bude tedy rovna

$$Y = \{y_1, y_2, y_3, ..., y_N\} = \{x_1^1, y_1^1, y_1^2, x_2^1, y_2^1, y_2^2, ..., x_N^1, y_N^1, y_N^2\}, \qquad (1.1)$$

kde N je délka vstupní nezabezpečené posloupnosti *u*. Pokud je blok děrování použit a je nastaven například tak, aby vybíral jenom liché paritní bity z prvního a sudé paritní bity z druhého RSC kodéru, výsledná informační rychlost bude $R_v = 1/2$ a v čase *k* bude výstup turbokodéru $y_k = (x_k^1, y_k^p)$. Celkový výstup z turbokodéru bude v tomto případě

$$Y = \{y_1, y_2, y_3, ..., y_N\} = \{x_1^1, y_1^1, x_2^1, y_2^2, x_3^1, y_3^1, ..., x_N^1, y_N^z\}, \quad (1.2)$$

přičemž $z = 1$, pokud N bude liché číslo a naopak $z = 2$, pokud N bude číslo sudé.

1.2 Nerekurzivní konvoluční kodéry (NRC)

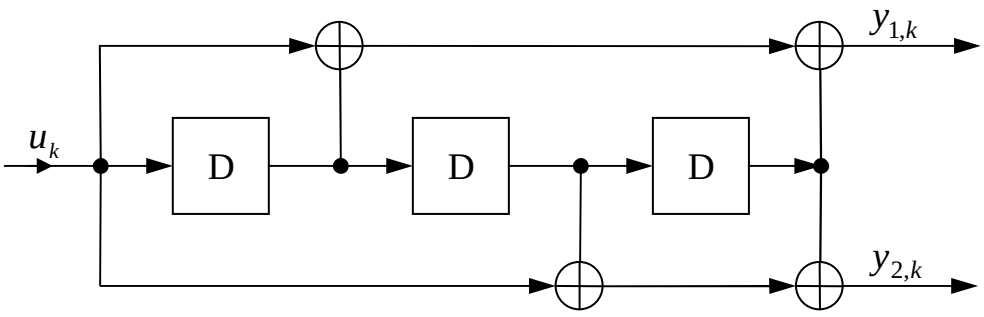
Konvoluční kodéry můžeme rozdělit do dvou základních skupin. První skupinu tvoří klasické nerekurzivní konvoluční kodéry (NRC). Základním prvkem NRC kodéru je posuvný registr složený z několika paměťových buněk, které v sobě uchovávají předchozí hodnoty vstupního bitového toku. Jednotlivé výstupy NRC kodéru jsou tvořeny lineární kombinací současných a předchozích vstupních hodnot. Na obrázku 1.2 můžeme vidět příklad zapojení NRC kodéru s informační rychlostí $R = 1/2$ a délkou kódového ohraničení $K = 4$. Délka kódového ohraničení udává, jak dlouho se jednotlivý bit ze vstupní nezabezpečené posloupnosti podílí na zabezpečovacím procesu a pro její výpočet platí vztah

$$K = m + 1, \quad (1.1.1)$$

kde *m* značí počet paměťových buněk kodéru. Toto zapojení je nesystematické, což znamená, že vstupní data nejsou přímo odesílána na některý z paritních výstupů. Obecně ale mohou být NRC kodéry konstruovány jako systematické i jako nesystematické. Tento konvoluční kodér můžeme popsat také pomocí generující matice, která bude mít v našem případě tvar

$$\mathbf{G} = [G_1(D), G_2(D)], \quad (1.1.2)$$

kde $G_1(D) = 1 + D + D^3$ a $G_2(D) = 1 + D^2 + D^3$.



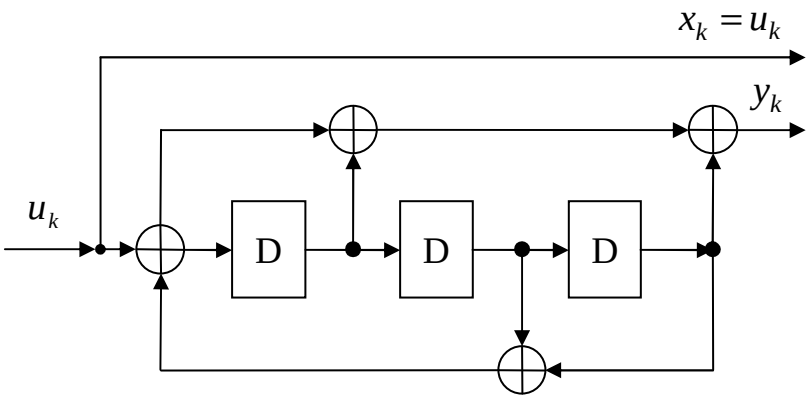
Obr. 1.2: Zapojení NRC kodéru s $R = 1/2$ a $K = 4$.

1.3 Rekurzivní systematické konvoluční kodéry (RSC)

RSC kodér získáme z klasického NRC kodéru tak, že jeden z jeho zabezpečovacích výstupů zavedeme zpět na jeho vstup jako zpětnovazební smyčku a druhý výstup nastavíme tak, aby poskytoval systematický výstup ($x_k = u_k$). Obrázek 1.3 ukazuje zapojení RSC kodéru odvozené z NRC kodéru na předchozím obrázku. Je patrné, že výstup NRC kodéru, reprezentovaný generujícím mnohočlenem $G_2(D)$, byl zaveden zpět na vstup RSC kodéru. Generující matice takto vzniklého RSC kodéru bude mít tvar

$$\mathbf{G} = \left[1, \frac{G_1(D)}{G_2(D)}\right], \quad (1.2.3)$$

kde $G_1(D)$ a $G_2(D)$ jsou generující mnohočleny NRC kodéru uvedeného výše a 1 značí přítomnost systematického výstupu.

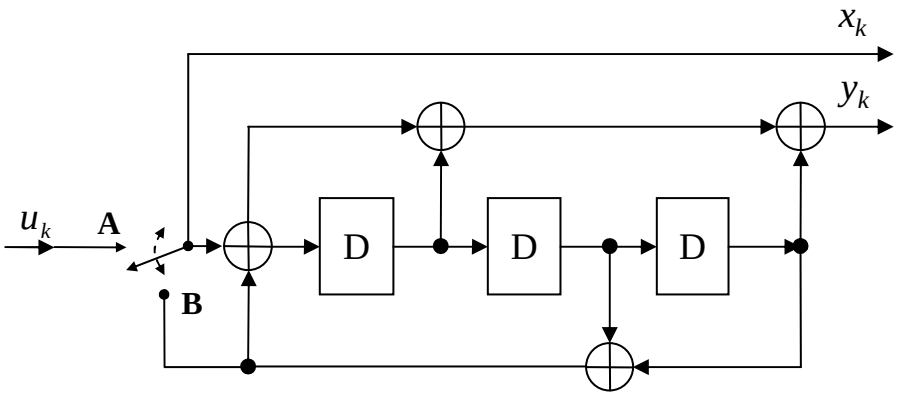


Obr. 1.3: Zapojení RSC kodéru odvozené z odpovídajícího NRC kodéru.

1.4 Ukončovací bity

Kvůli zjednodušení a zlepšení vlastností dekódovacího procesu se vstupní kódové slovo doplňuje o tzv. ukončovací bity (tail bits). Úkolem těchto přídavných bitů je uvést daný kodér do takového stavu, kdy po průchodu takto upraveného vstupního kódového slova kodérem budou všechny jeho paměťové buňky vynulovány [10].

U klasických konvolučních kodérů toho docílíme tak, že k danému kódovému slovu jednoduše přidáme *m* dodatečných nulových bitů, přičemž *m* je počet paměťových buněk kodéru. Tím bude zajištěno, že před vstupem každého kódového slova, i po jeho průchodu kodérem budou všechny jeho paměťové buňky vynulovány. Tento způsob ovšem není možné použít u RSC kodérů díky jejich zpětnovazební smyčce. Hodnoty jednotlivých ukončovacích bitů závisí na konkrétním stavu kodéru a je velmi těžké je předem předpovídat. I kdybychom dokázali u jednoho z kodérů tyto ukončovací bity předem najít, druhý RSC kodér by nemusel být (přidáním stejných ukončovacích bitů ke kódovému slovu) přiveden do nulového stavu, protože v bloku prokládání dojde k přeskládání jednotlivých vstupních bitů a tím se změní stavy kodéru při průchodu tohoto kódového slova. Obrázek 1.4 ukazuje jednoduché řešení, které odstraňuje problémy při hledání ukončovacích bitů u RSC kodérů.



Obr. 1.4: Zapojení RSC kodéru umožňující vynulování paměťových buněk.

Při průchodu vstupního kódového slova je přepínač ve stavu A, poté je přepínač přepnut do pozice B a posuvný registr provede dalších *m* kroků. Tímto způsobem bude ke vstupní nezabezpečené posloupnosti přidáno *m* zabezpečovacích bitů (v našem případě 3 bity), jejichž hodnoty přivedou paměťové buňky daného RSC kodéru do nulového stavu.

1.5 Zabezpečovací schopnosti kódů

Hammingova vzdálenost d udává počet míst (bitů), ve kterých se dvě různá kódová slova liší. Minimální Hammingova vzdálenost kódu je nejmenší Hammingova vzdálenost z množiny všech Hammingových vzdáleností mezi jednotlivými kódovými slovy. Pro $d_{\min}$ platí vztah

$$d_{\min} = \min\{w(x_i \oplus x_j)\} \quad i \neq j, \tag{1.4.1}$$

kde x_i a x_j jsou kódová slova a $w(x)$ je Hammingova váha, která vyjadřuje počet nenulových prvků v kódovém slově. Symbol $\oplus$ představuje operaci exklusivního součtu (XOR). Pro lineární kódy platí, že pokud sečteme dvě různá kódová slova, dostaneme jako výsledek další kódové slovo, které patří do množiny všech možných kódových slov [2]. V tomto specifickém případě (lineární kódy) tedy platí

$$d_{\min} = \min\{w(x_i)\}, \tag{1.4.2}$$

přičemž x_i je libovolné kódové slovo, s výjimkou kódového slova obsahující samé nuly. Pro nalezení $d_{\min}$ v případě lineárních kódů je tedy nutné najít nejmenší Hammingovu váhu ze všech možných kódových slov.

Obecně můžeme zabezpečovací schopnost určitého kódu vyjádřit pomocí následujících vztahů:

Smíšená schopnost ….. $d_{\min} = p_D + p_K + 1.$ (1.4.3)

Detekční schopnost ….. $d_{\min} = p_D + 1.$ (1.4.4)

Korekční schopnost ….. $d_{\min} = 2p_K + 1.$ (1.4.5)

V předchozích rovnicích p_D označuje počet detekovaných chyb a p_K počet korigovaných chyb. Pokud tedy známe minimální Hammingovu vzdálenost kódu, můžeme pomocí rovnice (1.4.6) zjistit, kolik chyb bude daný kód maximálně schopný opravit.

$$d_{\min} = 2p_K + 1 \Rightarrow p_K = \frac{d_{\min} - 1}{2}. \tag{1.4.6}$$

Turbokódy můžeme zařadit do skupiny lineárních systematických blokových kódů, což je nejvíce patrné ze struktury jejich výstupní zabezpečené posloupnosti (viz vztah 1.1). Pro výpočet jejich $d_{\min}$ tedy platí vztah 1.4.2.

1.6 Podrobnější rozbor vlastností RSC kodérů

RSC kodéry jsou základním stavebním prvkem turbokódů a zvyšují jejich celkovou výkonnost. Mezi jejich základní vlastnosti patří:

1.6.1 Snadná realizace

RSC kodér získáme z klasického NRC kodéru tak, že jeden z jeho zabezpečovacích výstupů zavedeme zpět na jeho vstup jako zpětnovazební smyčku a druhý výstup nastavíme tak, aby poskytoval systematický výstup (viz část 1.2).

1.6.2 Systematičnost

Použití systematických kodérů je výhodné z toho důvodu, že díky nim dochází ke zvýšení výsledné informační rychlosti turbokodéru, protože je potřeba přenášet systematická data pouze z jednoho konvolučního kodéru. Obrázek 1.5 ukazuje zapojení

turbokodéru, který se skládá ze dvou RSC kodérů s $R = 1/2$ vzájemně oddělených blokem prokládání. Je vidět, že nezabezpečená vstupní posloupnost přicházející do turbokodéru, je přes systematický výstup prvního RSC kodéru přivedena přímo na výstup turbokodéru. Tato vstupní posloupnost je ale také současně přivedena do bloku prokládání, kde je převedena do nového (nejčastěji pseudonáhodného) tvaru a takto upravená posloupnost dále přichází na vstup druhého RSC kodéru. Z výše uvedeného vyplývá, že systematická část druhého RSC kodéru je jenom „zpřeházenou“ (podle pravidel určených pro blok prokládání) verzí systematické posloupnosti prvního RSC kodéru. Pokud tedy použijeme v turbodekodéru stejný blok prokládání jako v případě turbokodéru, je možné systematickou posloupnost druhého RSC kodéru odvodit z přijaté systematické části prvního RSC kodéru a není nutno ji samostatně přenášet. Tímto je dosaženo zvýšení informační rychlosti na $R = 1/3$, oproti informační rychlosti $R = 1/4$, kterou bychom získali použitím nesystematických konvolučních kodéru. Toto navýšení informační rychlosti je ale až vedlejší výhodou RSC kodérů. Jejich hlavním důvodem použití v turbokódech je jejich rekurzivní povaha.

1.6.3 Rekurzivní povaha

Pro danou vstupní datovou posloupnost mají RSC kodéry tendenci generovat kódová slova s větší Hammingovou váhou než NRC kodéry, ze kterých byly tyto RSC kodéry odvozeny. Výsledkem je menší počet kódových slov s malou Hammingovou váhou a tím i zlepšení zabezpečovací schopnosti navrhovaného turbokódu. Přesto se ale může vyskytnout na vstupu RSC kodéru taková datová sekvence, která bude na jeho výstupu generovat kódové slovo s malou Hammingovu váhou. Rozložení Hammingovy váhy ve výstupním kódovém slově turbokodéru závisí na kombinaci kódových slov ze zabezpečovacích výstupů obou RSC kodérů. Snažíme se tedy zabránit výskytu takových kombinací, kdy by oba RSC kodéry generovaly na svých paritních výstupech kódová slova s malou Hammingovou váhou. Výskytu těchto nechtěných kombinací může být ve většině případů zabráněno správným návrhem bloku prokládání (podrobněji bude rozebráno v části 1.7). Prokladače, které prohazují pořadí jednotlivých bitů náhodným nebo pseudonáhodným způsobem, poskytují v tomto smyslu lepší výsledky než klasické blokové prokladače.

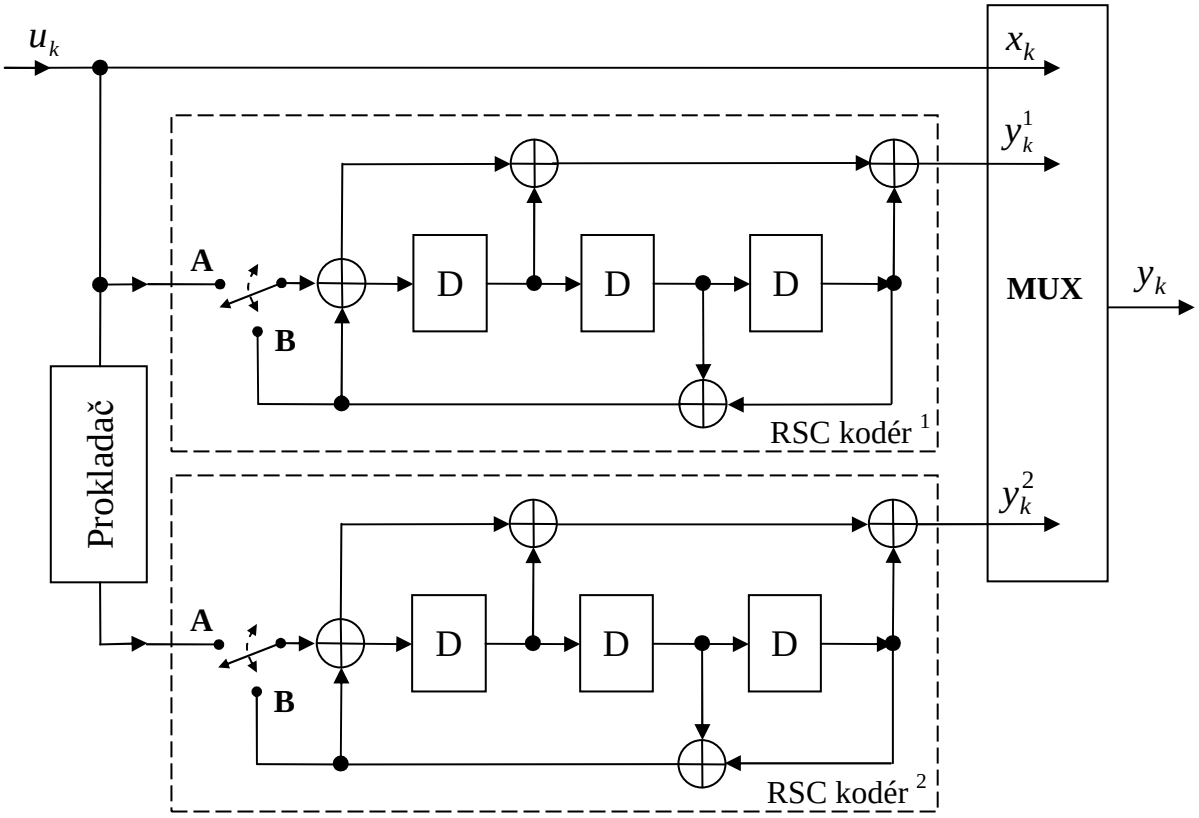
1.6.4 Nekonečná impulsní odezva IIR (infinite impulse response)

Kodér se zavedenou zpětnovazební smyčkou generuje rekurzivní kód, který má nekonečnou impulsní odezvu IIR, zatímco pro kodér bez zpětné vazby (klasické nerekurzivní kodéry) je charakteristická konečná impulsní odezva FIR.

Pokud bychom v zapojení turbokodéru z obrázku 1.5 namísto RSC kodérů použili NRC kodéry, potom by vstupní posloupnost s Hammingovou váhou rovnou jedné (například $u = (000..010..000)$ generovala na výstupu prvního NRC kodéru, díky jeho konečné impulsní odezvě (FIR), kódové slovo s velmi malou Hammingovou váhou. Jelikož prokladač nemění Hammingovu váhu vstupní posloupnosti (dochází v něm jenom k přeskupení jednotlivých bitů vstupní sekvence), i na výstupu druhého NRC se objeví posloupnost s velmi malou Hammingovou váhou (stejnou jako v případě prvního NRC kodéru). Z toho plyne, že pro jakoukoli vstupní posloupnost s touto nejhorší možnou Hammingovu váhou bude, díky konečné impulsní odezvě NRC kodéru, Hammingova váha výsledné kódové posloupnosti turbokodéru vždy velmi malá. Tím se zmenší i minimální Hammingova vzdálenost $d_{\min}$ a klesne celková zabezpečovací schopnost celého turbokódu.

Když naopak použijeme, stejně jako na obrázku 1.5, RSC kodéry a budeme opět uvažovat vstupní posloupnost s nejnižší možnou Hammingovou váhou, bude tato posloupnost na výstupu RSC kodéru generovat nekonečnou impulsní odezvu IIR. Pro nekonečně dlouhou vstupní posloupnost s touto Hammingovou váhou by na výstupu byla

generována posloupnost s nekonečnou Hammingovou váhou. V praktických případech jsme samozřejmě vždy omezeni konečnou délku, a proto i výsledná Hammingova váha není nikdy nekonečná. Z této podstatné vlastnosti RSC kodérů plyne, že v případě vstupního kódového slova s Hammingovou váhou rovnou jedné, budou oba RSC kodéry na svých výstupech generovat kódová slova s velkou Hammingovou váhou.



Obr. 1.5: Zapojení turbokodéru s $R = 1/3$.

1.7 Blok prokládání

Nejčastěji se prokládání používá jako doplněk ke kanálovému kódování v případě, že chceme chránit přenášená zabezpečená data před shlukem chyb, protože při výskytu většího počtu chyb by mohla být překročena zabezpečovací schopnost daného kódu. Díky prokládání je shluková chyba rozprostřena do více přenášených datových bloků a díky tomu není zabezpečovací schopnost daného kódu překročena. Zabezpečení přenášených dat před shlukovými chybami ovšem není hlavním důvodem použití prokladačů v turbokódech. Prokladač v případě turbokódů tedy není umístěn na samotný konec celého kódovacího procesu, ale je umístěn mezi dva RSC kodéry.

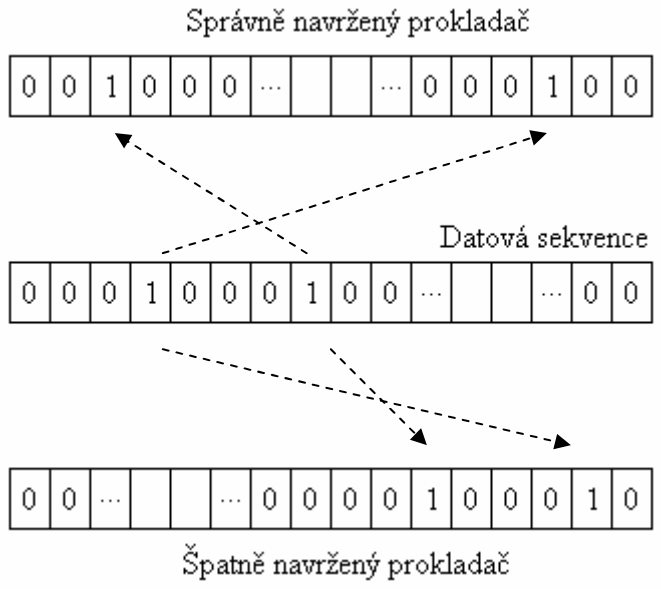
Návrh vhodného prokladače je klíčovým faktorem, který ovlivňuje celkovou výkonnost turbokódů. V jejich případě je hlavním úkolem prokladačů zajistit, společně s RSC kodéry, aby Hammingova váha výsledného kódové slova byla velká i v případech, kdy se na vstupu turbokodéru objeví některá z *“ nejméně vhodných “* datových sekvencí, mezi které patří i již zmiňovaná vstupní datová slova s Hammingovou váhou rovnou jedné. Jak již bylo popsáno dříve, tato datová slova nesnižují zabezpečovací schopnost daného turbokódu, a to díky nekonečné impulsní odezvě RSC kodérů. Vstupní bit s hodnotou 1 sice dostane RSC kodér z nulového stavu, ale sám se již (díky IIR) nikdy do nulového stavu nevrátí. Do nulového stavu tento kodér vrátíme jenom díky přidaným ukončovacím bitům.

Pro rekurzivní kodéry dále obecně platí, že výstupní kódové slovo s nejmenší Hammingovou váhou dostaneme, pokud se ve vstupní datové posloupnosti (složené z delší posloupnosti nul) objeví bitová sekvence odpovídající generujícímu mnohočlenu $G_2(D)$ z rovnice (1.1.1). Připomeňme, že $G_2(D)$ je generující mnohočlen výstupu původního NRC kodéru, který byl zaveden jako zpětnovazební smyčka zpět na vstup, díky čemuž jsme získali zapojení RSC kodéru (obr. 1.3). V našem případě je $G_2(D)=1+D^2+D^3$, a proto kódové slovo s nejmenší Hammingovou váhou se na výstupu RSC kodéru objeví, pokud vstupní posloupnost bude rovna například $u=(000....00101100)$. Další nevhodné vzory pro vstupní datová slova jsou některá slova s Hammingovou váhou rovnou dvěma, kdy se v této sekvenci vyskytují dva jedničkové bity oddělené určitým počtem nulových bitů. Může dojít k situaci, kdy první jedničkový bit sice dostane kodér z nulového stavu, ale nevhodně umístěný druhý jedničkový bit může tento kodér opět rychle vrátit zpět do nulového stavu. Všechna tato nejméně vhodná datová slova se obecně označují jako samo-ukončovací sekvence, což nejlépe charakterizuje jejich povahu. V případě, že se na vstupu turbokodéru objeví některá samo-ukončovací sekvence, automaticky dostáváme na paritním výstupu prvního RSC kodéru kódové slovo s malou Hammingovou váhou. V této situaci může Hammingovu váhu výstupního kódového slova turbokodéru zvýšit pouze paritní výstup generovaný ve druhém RSC kodéru.

Zde vstupuje do kódovacího procesu blok prokládání, jehož hlavní význam se stává patrným právě v situaci, kdy je potřeba jednotlivé bity vstupní samo-ukončovací sekvence přeskládat do nového tvaru, který již nebude mít samo-ukončovací charakter. K tomuto účelu se nejčastěji používají náhodné nebo pseudonáhodné prokladače, které v tomto smyslu vykazují lepší vlastnosti jako obvyklé blokové prokladače. Takový prokladač tedy nemění Hammingovu váhu vstupní datové posloupnosti, pouze dochází k přeskládání jednotlivých bitů. Nejjednodušší možností je realizace pseudonáhodného prokladače pomocí mapovací tabulky (vhodné pro malé délky vstupního bloku dat). Další způsob realizace může být pomocí generátoru pseudonáhodné posloupnosti (například skrambler), kdy je vstupní sekvence převedena na novou pseudonáhodnou posloupnost.

Obrázek 1.6 ukazuje dvě možné situace, které mohou nastat v případě, kdy se na vstupu turbokodéru objeví některá ze samo-ukončovacích sekvencí. V příkladu uvedeném na předcházejícím obrázku jde o samo-ukončovací sekvenci $u=\{10001\}$. Je vidět, že správně navržený prokladač dokáže tuto sekvenci přeměnit na takovou, která bude generovat na výstupu RSC kodéru sekvenci s velkou Hammingovou váhou. Naopak špatný návrh prokladače povede jenom k zopakování této samo-ukončovací sekvence. Stejně tak může u špatně navrženého prokladače dojít k tomu, že bude samo-ukončovací sekvence $u=\{10001\}$ převedena na novou sekvenci, která bude mít ovšem taky samo-ukončovací charakter. Tím dojde k výraznému zhoršení zabezpečovacích schopností daného turbokódu.

Výsledkem vhodně navrženého prokladače je tedy velká Hammingova váha výstupního kódového slova i v těch nejhorších možných případech vstupní datové posloupnosti. Takto máme zajištěnu velkou hodnotu $d_{\min}$ a tím i vysokou zabezpečovací schopnost turbokódů.



Obr. 1.6: Ukázka možných způsobů návrhu prokladače.

1.8 Děrování

Děrování slouží k redukci počtu bitů ve výstupním kódovém slově turbokodéru a tím i šířky pásma potřebné k přenosu zakódované informace. Jeho nevýhodou ovšem je snížení celkové výkonnosti daného turbokódu, proto je nutné najít určitý kompromis mezi konečnou délkou kódového slova a požadovanou zabezpečovací schopností turbokódu.

Při použití dvou RSC kodéru s informační rychlostí $R = 1/2$ bude délka kódového slova na výstupu turbokodéru třikrát větší než délka vstupní datové sekvence [11]. Různých informačních rychlostí můžeme ovšem dosáhnout děrováním paritních bitových posloupností y_k^1 a y_k^2. Obrázek 1.7 ukazuje proces děrování na dvou příkladech. Hodnota 1 v uvedených tabulkách reprezentuje bit, který je zahrnut do výstupní přenášené bitové sekvence, naopak hodnota 0 reprezentuje vyřazený bit. U každého děrovacího vzoru je vidět i část výstupní posloupnosti turbokodéru.

x_k^1	1	1	1	1	1	1	1
y_k^1	1	0	1	0	1	0	1
y_k^2	0	1	0	1	0	1	0

→

x_1^1	y_1^1	x_2^1	y_2^2	x_3^1	y_3^1	x_4^1

x_k^1	1	1	1	1	1	1	1
y_k^1	1	0	0	0	0	0	1
y_k^2	0	0	0	1	0	0	0

→

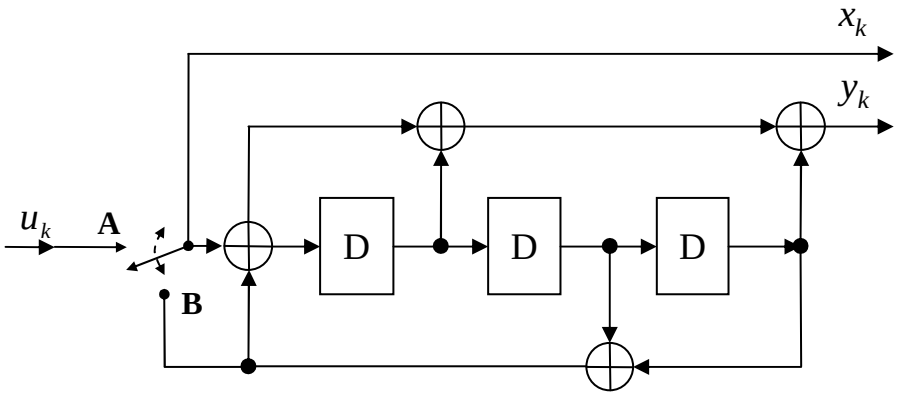
x_1^1	y_1^1	x_2^1	x_3^1	x_4^1	x_4^2	x_5^1

Obr. 1.7: Děrovací vzory pro informační rychlosti $R = 1/2$, $R = 3/4$

2 Dekódování turbokódů

2.1 Dekódování konvolučních kódů – Viterbi algoritmus

Konvoluční kodéry jsou základní součástí turbokódů, proto zde chci uvést příklad dekódování jednoduchého konvolučního kódu. Zapojení kodéru je uvedeno na následujícím obrázku.



Obr. 2.1: Zapojení RSC kodéru.

Jedná se o konvoluční kód s délkou kódového ohraničení $K = 4$ a informační rychlostí $R = 1/2$. Z toho vyplývá, že každý vstupní bit tohoto kodéru bude během zabezpečovacího procesu ovlivňovat 4 páry výstupních bitů. Následující tabulka nám umožňuje popsat všechny možné stavy konvolučního kodéru z našeho příkladu.

Stav paměti	Vstup	Následující stav	Výstup	
D1-D2-D3	U_k	D1-D2-D3	X_1	X_2
000	1	100	1	1
000	0	000	0	0
001	1	000	1	1
001	0	100	0	0
010	1	001	1	0
010	0	101	0	1
011	1	101	1	0
011	0	001	0	1
100	1	110	1	0
100	0	010	0	1
101	1	010	1	0
101	0	110	0	1
110	1	011	1	1
110	0	111	0	0
111	1	111	1	1
111	0	011	0	0

Tab. 2.1: Tabulka popisující jednotlivé stavy konvolučního kodéru.

Uvažujme vstupní datovou posloupnost o délce 10 bitů {1011001101}. Výstupní sekvence bude podle tabulky 2.1 rovna {11 01 10 11 00 00 11 10 00 11}. Dále je nutné k naší vstupní bitové posloupnosti přidat ještě další 3 ukončovací bity {001}, díky nimž uvedeme

všechny paměťové buňky kodéru po skončení zabezpečovacího procesu do nulového stavu. Konečná výstupní sekvence bude tedy vypadat následovně

{11 01 10 11 00 00 11 10 00 11 00 01 11}. (2.1.1)

Budeme předpokládat, že během přenosu této zabezpečené posloupnosti došlo ke dvěma chybám a na vstup dekodéru přišla posloupnost

{11 01 10 11 00 0**1** 11 10 00 **0**1 00 01 11}, (2.1.2)

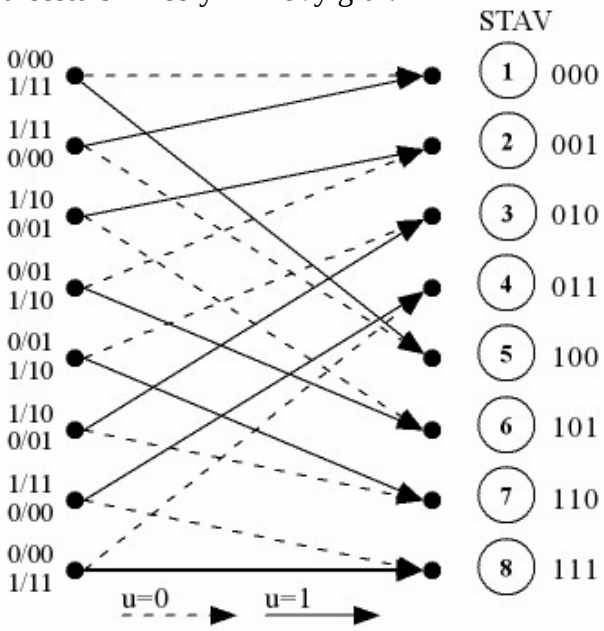
ve které jsou tučně vyznačeny chybně přijaté bity.
Viterbi algoritmus dekódování se dá názorně ukázat na mřížovém grafu daného kódu. Pro náš příklad bude mít mřížový diagram podobu uvedenou na obrázku 2.3.

Jednotlivé uzly mřížového grafu odpovídají stavu paměti kodéru při zabezpečovacím procesu. Je vidět, že kodér začíná i končí ve stavu (000). Přechody znázorněné plnou čarou odpovídají situaci, kdy se na vstupu kodéru objevil bit s hodnotou 1 a naopak přerušovaná čára odpovídá vstupní hodnotě 0. Tučnou čárou je znázorněn průchod mřížovým grafem pro naši vstupní posloupnost. Jednotlivé přechody v mřížovém grafu jsou opět odvozeny z tabulky 2.1. Obrázek 2.2 ukazuje detailní podobu jednoho stavově-časového kroku tohoto mřížového grafu.

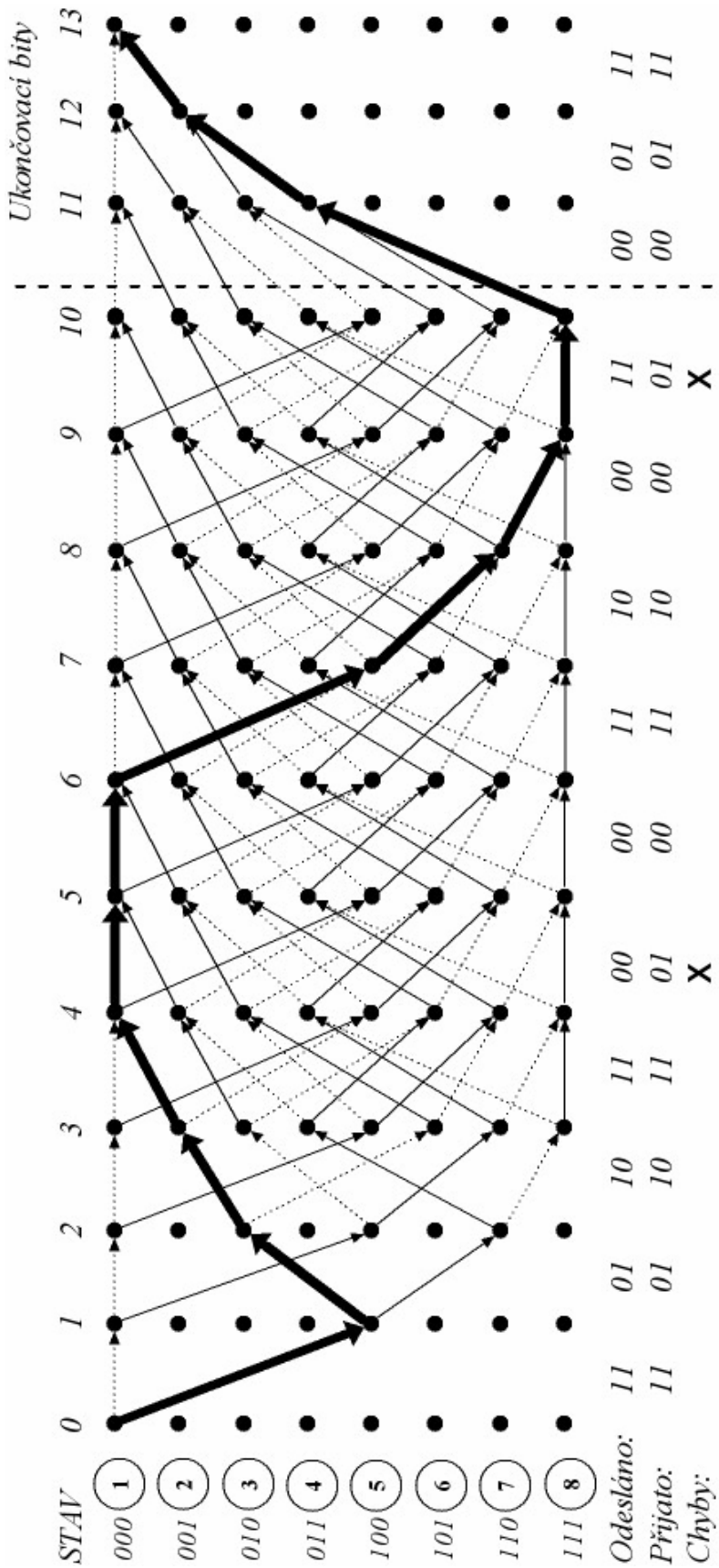
Při dekódování pomocí Viterbi algoritmu se v každém úseku mřížového diagramu spočítá Hammingova vzdálenost mezi přijatým párem bitů a všemi možnými dvojicemi bitů, které bychom mohli přijmout [2, 5]. Je tedy patrné, že se k výběru správné cesty skrz mřížový graf přibližujeme postupně a jako metrika ke konečnému rozhodnutí o volbě této cesty mřížovým grafem se používá akumulovaná Hammingova vzdálenost d_A. Pro výpočet aktuální hodnoty d_A, vztažené k určitému stavu, platí vztah

aktuální d_A = předchozí d_A + aktuální Hammingova vzdálenost dvojic bitů. (2.1.3)

V každém úseku mřížového grafu spočítáme všechny hodnoty d_A a jako nejpravděpodobnější cestu zvolíme přechod do stavu, který bude mít nejnižší hodnotu d_A. Takto postupujeme po jednotlivých úsecích až na konec mřížového grafu, čímž získáme nejvíce pravděpodobnou cestu skrz celý mřížový graf.



Obr. 2.2: Jeden stavově-časový krok mřížového graf.



Obr. 2.3: Mřížový graf pro konvoluční kód z obrázku 2.1.

Při výpočtu d_A v jednotlivých úsecích mřížového grafu mohou nastat následující problémy:

1) Na obrázku 2.3 je vidět, že od úseku číslo 4 dostáváme dvě hodnoty akumulované vzdálenosti d_A vztažené k jednomu stavu paměti. S těmito dvěma hodnotami d_A pro každý stav se vypořádáme tím způsobem, že obě hodnoty porovnáme a dále pracujeme jenom s hodnotou nižší. Vyšší hodnota je vyřazena z procesu dekódování.
2) Druhý problém nastává v okamžiku, kdy dostáváme ve dvou nebo více stavech najednou stejnou (nejnižší) hodnotu d_A. Tento problém řešíme náhodným výběrem dalšího přechodu nebo determinovaným výběrem podle nějakého stanoveného pravidla (například vybereme vždy tu větev, která vede směrem dolů). Mezi oběma způsoby není rozdíl a je tedy jedno, který z nich použijeme.

KROK	0	1	2	3	4	5	6	7	8	9	10	11	12	13
STAV 1	**0**	2	3	4	**0**	**1**	**1**	3	4	4	5	5	4	**2**
STAV 2				**0**	4	3	3	3	4	4	3	3	**2**	
STAV 3			**0**	5	3	2	2	4	3	3	4	4		
STAV 4				3	3	4	4	2	5	3	**2**	**2**		
STAV 5		**0**	3	4	2	**1**	3	**1**	4	4	5			
STAV 6				2	4	3	3	3	2	4	3			
STAV 7			2	3	3	4	2	4	**1**	3	4			
STAV 8				3	3	4	4	4	5	**1**	**2**			

Tab. 2.2: Tabulka hodnot d_A pro všech 13 úseků mřížového grafu.

Předchozí tabulka ukazuje hodnoty d_A pro všech 13 úseků mřížového grafu z našeho příkladu. Můžeme si všimnout, že nejnižší číslo v každém sloupci tabulky odpovídá počtu chyb vzniklých při přenosu. Po výpočtu všech hodnot v tabulce 2.2 pokračuje dekódovací proces následovně:

1) Z posledního sloupce tabulky 2.2 vybereme stav s nejnižší hodnotou d_A a zapamatujeme si číslo tohoto stavu.
2) Budeme zpětně procházet tabulkou 2.2 až na její začátek a pro každý stav vybereme předchozí stav s nejnižší hodnotou d_A . (Pro zjištění předcházejících stavů k jakémukoliv zapamatovanému stavu můžeme použít například obrázek 2.3, kde jsou přehledně znázorněny jednotlivé přechody mezi stavy.) Tento proces se označuje jako zpětné trasování a dochází při něm k opravě chyb. Například v úseku číslo 5 z tabulky 2.2 je vidět, že v dopředném směru dochází k nejednoznačnosti při volbě přechodu, protože máme minimální hodnotou d_A ohodnoceny dva stavy současně. Kdybychom tedy při náhodném výběru další cesty zvolili v dopředném směru chybný přechod, při zpětném trasování by tato chybná volba byla odhalena. To je patrné při pohledu na obrázek 2.3, kdy ze stavu číslo 1 v šestém úseku mřížového grafu nevede žádný zpětný přechod do stavu 5, ale pouze do stavu 1 a 2. Následující tabulka ukazuje zapamatované stavy, získané při zpětném trasování.

KROK	0	1	2	3	4	5	6	7	8	9	10	11	12	13
STAV	**1**	**5**	**3**	**2**	**1**	**1**	**1**	**5**	**7**	**8**	**8**	**4**	**2**	**1**

Tab. 2.3: Zapamatované stavy při zpětném trasování.

Následně zbývá procházet v dopředném směru tabulkou zapamatovaných stavů a pomocí obrázku 2.3 zjistit, který vstupní bit odpovídá danému přechodu z předchozího stavu do stavu následujícího. Tím získáme vstupní bitovou posloupnost, která byla kódována konvolučním kodérem.

	Vstupní bitová sekvence										Ukončovací bity		
KROK	1	2	3	4	5	6	7	8	9	10	11	12	13
HODNOTA BITU	1	0	1	1	0	0	1	1	0	1	0	0	1

Tab. 2.4: Dekódovaná vstupní posloupnost.

Kvůli narůstajícímu zpoždění a velikosti paměťového prostoru se optimální délka pro zpětné trasování udává jako $5*K$.

2.2 Dekódovací náročnost pro konvoluční kódy

U většiny konvolučních kódů se vstupní informační posloupnost skládá z $r*T$ bitů, kde r udává počet paralelních informačních bitů v jednom časovém intervalu a T je počet těchto časových intervalů [2]. Tím dostáváme $T+m$ (m je počet paměťových buněk kodéru) úseků v mřížovém grafu a počet různých cest tímto grafem je 2^{r*T}. Hledání nejpravděpodobnější cesty v případě postupného pravděpodobnostního dekódování, kdy srovnáváme přijatou posloupnost se všemi možnými, má tedy výpočetní náročnost

$$R = \left\{2^{r*T}\right\}. \tag{2.2.1}$$

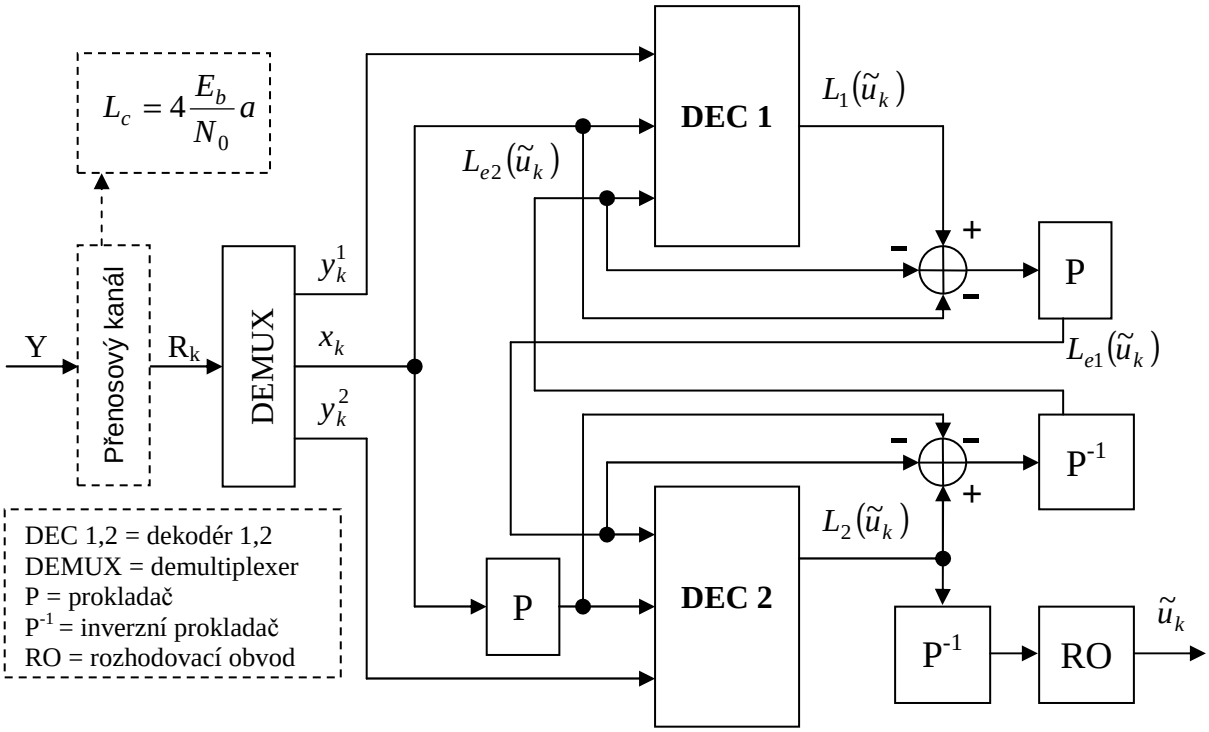
Viterbi algoritmus redukuje tuto výpočetní náročnost tak, že rozhoduje o nejpravděpodobnější cestě v jednotlivých úsecích mřížového grafu zvlášť. Počet uzlů (stavů) na jeden úsek mřížového grafu je 2^m a na každý jednotlivý uzel připadá 2^r výpočtů. Výpočetní náročnost Viterbi algoritmu je dána vztahem

$$R = \left\{\left(2^m\right)\left(2^r\right)\left(T+m\right)\right\}. \tag{2.2.2}$$

Z rovnice (2.2.2) vyplývá, že zvyšováním T narůstá výpočetní náročnost u Viterbi algoritmu lineárně, což vede k výraznému snížení množství výpočetních operací nutných k nalezení nejpravděpodobnějšího průchodu mřížovým grafem. Výpočetní náročnost u Viterbi algoritmu naopak poroste exponenciálně, pokud zvyšujeme parametr *r* nebo *m*.

2.3 Iterativní dekódování turbokódů

Obrázek 2.4 ukazuje blokové schéma dekodéru turbokódu. Vstupem do tohoto turbodekodéru je sekvence $R_k = \left\{x_k, y_k^1, y_k^2\right\}$ přijatá z přenosového kanálu. Parametr $L_c = 4\frac{E_b}{N_0}a$ vyjadřuje spolehlivost přenosového kanálu [4]. Tímto parametrem jsou například u SOVA algoritmu (soft-output Viterbi algoritmus) následně násobeny (váženy) jednotlivé vstupy turbodekodéru. Vstupní sekvence R_k je nejdříve na vstupu turbodekodéru demultiplexována na jednotlivé bitové sekvence. Základní struktura turbodekodéru, kdy jsou dva dílčí dekodéry odděleny blokem prokládání, je podobná struktuře turbokodéru z obr. 1.1.



Obr. 2.4: Obecné blokové schéma dekodéru turbokódu.

Na oba dekodéry přicházejí tři následující vstupy:

1) Vstupní systematická posloupnost $X = \{x_1, x_2, x_3, ..., x_N\}$.
2) Paritní posloupnost, která je určena pro daný dekodér. Pro první dekodér dostáváme tedy sekvenci $Y^1 = \{y_1^1, y_2^1, y_3^1, ..., y_N^1\}$ a pro druhý sekvenci $Y^2 = \{y_1^2, y_2^2, y_3^2, ..., y_N^2\}$.
3) Informace o pravděpodobných hodnotách dekódovaných bitů získané z druhého dílčího dekodéru. Tyto informace jsou označovány jako apriorní informace.

Z těchto vstupních hodnot následně oba dekodéry počítají tzv. měkký (soft) výstup $L(\tilde{u}_k)$. Měkký výstup doplňuje informaci o dekódované posloupnosti $\tilde{u} = \{\tilde{u}_1, \tilde{u}_2, \tilde{u}_3, ..., \tilde{u}_N\}$, narozdíl od klasického Viterbi algoritmu, i o míru spolehlivosti tohoto odhadu pro každý jednotlivý bit dekódované sekvence.

Proces dekódování pokračuje tím, že na výstupu daného dekodéru jsou od měkkého výstupu odečtena systematická data a apriorní informace. Tím je získána konečná informace $L_e(\tilde{u}_k)$ z tohoto dekodéru, která je přivedena na vstup druhého dekodéru jako apriorní informace. Iterativní dekódování je tedy založeno na výměně těchto konečných informací $L_e(\tilde{u}_k)$ mezi dílčími dekodéry a opětovném dekódování vstupních dat. V jednotlivých krocích iterativního dekódování proto dochází k upřesňování odhadu dekódované sekvence. Celkový počet kroků při dekódování závisí na oblasti použití daného turbokódu. Po skončení iterativního dekódování dostáváme na výstupu druhého dekodéru konečnou podobu dekódované sekvence $\tilde{u} = \{\tilde{u}_1, \tilde{u}_2, \tilde{u}_3, ..., \tilde{u}_N\}$. Následuje popis dvou základních algoritmů používaných pro dekódování turbokódů.

2.4 Soft - output Viterbi (SOVA) algoritmus

Tento algoritmus obsahuje dvě zásadní modifikace oproti klasickému Viterbi algoritmu, které umožňují jeho použití v dekodérech turbokódů:

1) Je použita jiná metrika pro výběr nejpravděpodobnější cesty mřížovým grafem, která při výpočtech bere v úvahu i apriorní informace.
2) Algoritmus je modifikován tak, aby poskytoval měkký výstup $L(\tilde{u}_k)$ pro každý dekódovaný bit $\tilde{u}_k$. (SOVA algoritmus bere v úvahu, v každém úseku mřížového grafu, i konkurenční cestu k vybrané nejpravděpodobnější cestě).

Dekódování pomocí SOVA algoritmu bude vysvětleno na stejném příkladu jako v případě Viterbi algoritmu. Zapojení kodéru je uvedeno na obrázku 2.1 a jeden stavově-časový krok mřížového grafu na obrázku 2.3. Posloupnost vstupující do RSC kodéru je $u_k = \{1001100\}$. Po průchodu této vstupní posloupnosti kodérem jsou přidány 3 ukončovací bity $\{110\}$ a výsledná zabezpečená sekvence je {11 01 01 10 10 00 01 10 11 00}. Při přenosu došlo ke dvěma chybám a na vstupu dekodéru byla přijata posloupnost

$$\{11\ \underline{\mathbf{1}}1\ 01\ 10\ 11\ 0\underline{\mathbf{1}}\ 01\ 10\ 11\ 00\}. \tag{2.4.1}$$

Detailní matematické odvození vzorců, použitých v následujícím textu, je možné najít v literatuře [4, 10]. Budou použita jenom konečná řešení těchto rovnic, upravená pro účely našeho textu (algoritmů SOVA a MAP).

Před začátkem procesu dekódování je nejprve nutné, pro potřeby následných výpočtů, přijatou sekvenci demultiplexovat a následně jednotlivé bity mapovat do {-1,+1} rozsahu. Toto mapování provedeme jednoduše tak, že nulové bity nahradíme hodnotou -1. Proces dekódování poté pokračuje následovně:

1) V jednotlivých úsecích mřížového grafu vypočteme akumulovanou hodnotu metriky pro všechny uzly. V případě SOVA algoritmu se pro výpočet akumulované metriky používá vztah

$$M_t^{(s)} = M_{t-1}^{(s)} + \frac{1}{2}u_t^{(s)}L_c y_{t,1}^{(s)} + \frac{1}{2}\sum_{j=2}^{N} x_{t,j}^{(s)}L_c y_{t,j}^{(s)} + \frac{1}{2}u_t^{(s)}L(u_t). \tag{2.4.2}$$

Pro náš případ můžeme tento obecný vzorec upravit do tvaru

$$M_t^{(s)} = M_{t-1}^{(s)} + \frac{1}{2}u_t^{(s)}L_c y_{t,1}^{(s)} + \frac{1}{2}x_{t,2}^{(s)}L_c y_{t,2}^{(s)} + \frac{1}{2}u_t^{(s)}L(u_t). \tag{2.4.3}$$

$M_t^{(s)}$ ….. je hodnota akumulované metriky pro přechod do stavu *s* v čase *t*.

$u_t^{(s)}$, $x_{t,2}^{(s)}$ ….. je vysílaná dvojice bitů (systematický a jeho paritní bit).

$y_{t,1}^{(s)}$, $y_{t,2}^{(s)}$ ….. je dvojice bitů přijatá z přenosového kanálu, korespondující s vysílanou dvojicí bitů.

$L(u_t)$ ….. je apriorní hodnota pro bit u_t, získaná z předchozího dekodéru. Pokud jde o první krok a první dekodér, je hodnota nastavena na nulu.

L_c ….. je parametr určující spolehlivost daného přenosového kanálu. Pro AWGN kanál platí vztah

$$L_c = 4\frac{E_b}{N_0}, \tag{2.4.4}$$

kde poměr E_b/N_0 vyjadřuje odstup signálu od šumu.

2) V případě úseků, kdy do jednoho uzlu (stavu) vedou dva přechody, uchováváme obě hodnoty akumulované metriky.

Uvažujme dvě cesty $\hat{c}_k^s$ a c_k^s směřující v čase k do jednoho stavu S_k, které mají metriku $M(\hat{c}_k^s)$ a $M(c_k^s)$. Jestliže vybereme cestu $\hat{c}_k^s$ jako přežívající, protože má vyšší metriku, můžeme definovat rozdíl metrik jednotlivých cest jako

$$\Delta_k^s = M(\hat{c}_k^s) - M(c_k^s). \tag{2.4.5}$$

Hodnota Δ_k^s vyjadřuje míru spolehlivosti toho, že jsme učinili správné rozhodnutí, když jsme jako přežívající označili cestu $\hat{c}_k^s$ a vyloučili jsme tím cestu c_k^s.

Tabulka 2.5 ukazuje vypočítané hodnoty akumulované metriky v jednotlivých úsecích mřížového grafu. Při výpočtu jsem pro jednoduchost volil $L_c = 4$.

3) Vyhledáme nejpravděpodobnější cestu mřížovým grafem. Postupujeme stejně jako v případě Viterbi algoritmu s tím rozdílem, že v každém úseku mřížového grafu se jako nejpravděpodobnější cesta volí ta, která má nejvyšší hodnotu akumulované metriky. Tímto způsobem tedy získáme nejpravděpodobnější cestu mřížovým grafem a tím i dekódovanou bitovou sekvenci $\tilde{u} = \{\tilde{u}_1, \tilde{u}_2, \tilde{u}_3, ..., \tilde{u}_N\}$.
4) Dále je nutné v každém úseku mřížového grafu najít konkurenční cestu k vybrané nejpravděpodobnější cestě a spočítat rozdíl Δ_k^s mezi akumulovanou metrikou těchto cest (viz. obr. 2.5).
5) Následuje nalezení tzv. update sekvence pro každý úsek. Tato update sekvence určuje, které bity $\tilde{u}_k$ by změnily svoji hodnotu, pokud bychom v tomto úseku vybrali konkurenční cestu namísto nejpravděpodobnější cesty. Hodnota 1 v update sekvenci signalizuje odlišnou hodnotu daného bitu, přičemž nejméně významný bit je uveden vpravo. Z obrázku 2.5 je vidět, že například update sekvence pro úsek 4 je (1101), což znamená, že při volbě konkurenční cesty v tomto úseku by se změnily hodnoty dekódovaných bitů $\tilde{u}_1$, $\tilde{u}_3$, $\tilde{u}_4$.
6) Pro každý bit $\tilde{u}_k$ zjistíme (pomocí update sekvencí), které konkurenční cesty by dávaly odlišnou hodnotu tohoto bitu. Zapamatujeme si čísla úseků, kde se tyto konkurenční cesty nacházejí. V tabulce 2.6 je pro každý bit $\tilde{u}_k$ vidět zapamatované úseky, které ovlivňují daný bit a zjistíme jejich Δ_k^s. Poté již jsme schopni určit měkký výstup pro jednotlivé bity $\tilde{u}_k$ pomocí vztahu

$$L(\tilde{u}_k) = \tilde{u}_k * \min \Delta_k^s. \tag{2.4.6}$$

Tímto způsobem nakonec získáme měkký výstup $L(\tilde{u})$ pro celou dekódovanou sekvenci, který nám umožňuje vyjádřit míru spolehlivosti odhadu této dekódované sekvence.

7) Měkký výstup pro odhadovaný bit $\tilde{u}_k$ může být rozložen [4] do tří odlišných částí, přičemž platí vztah

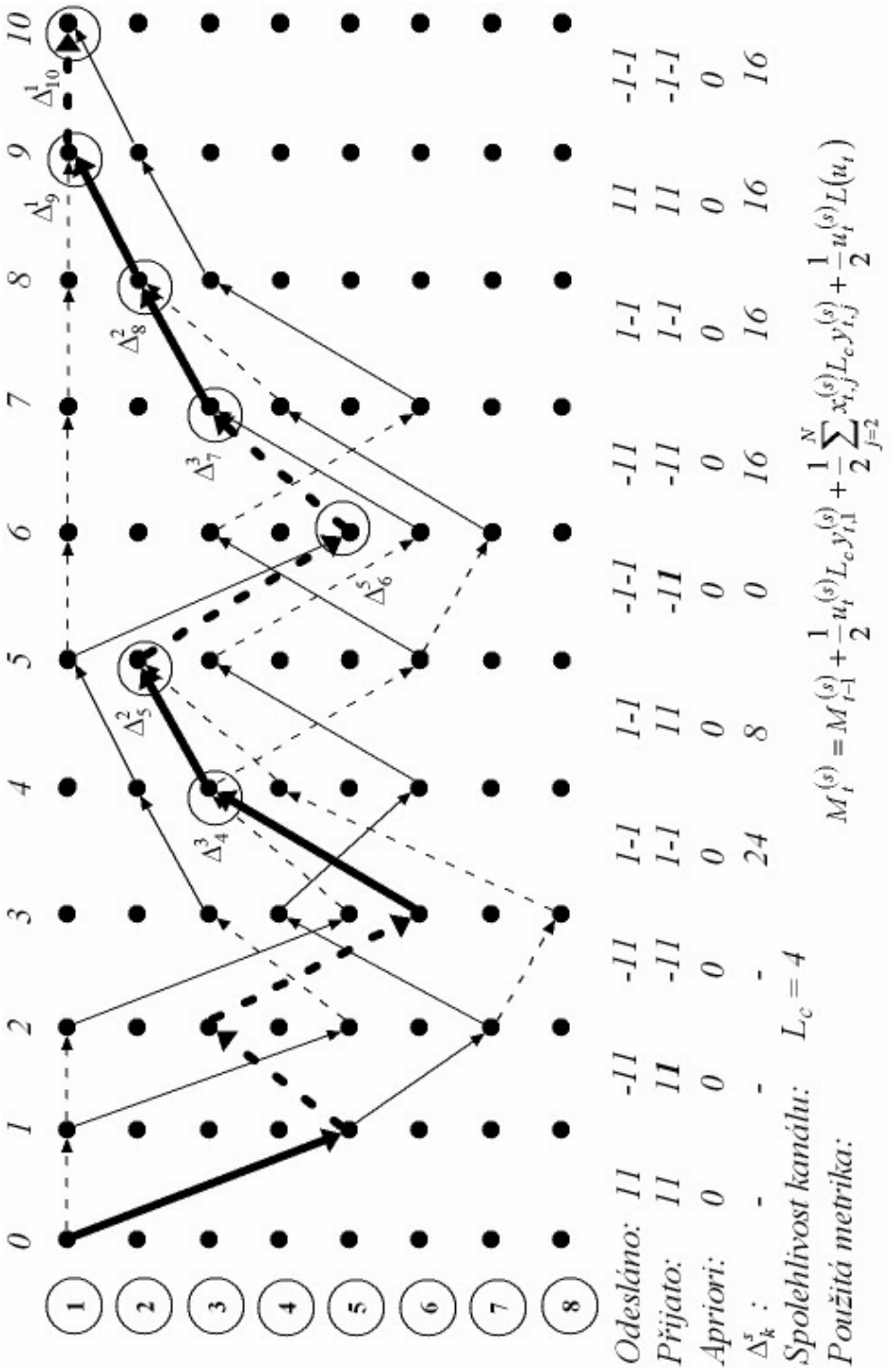
$$L(\tilde{u}_k) = L(u_k) + L_c y_{t,1} + L_e(\tilde{u}_k). \tag{2.4.7}$$

$L(u_k)$ ….. je apriorní hodnota získaná z předchozího dekodéru.

$L_c y_{t,1}$ ….. je systematická hodnota přijatá z přenosového kanálu, která je vynásobena parametrem L_c.

$L_e(\tilde{u}_k)$ …. je konečná hodnota produkovaná dílčím dekodérem a platí pro ni vztah

$$L_e(\tilde{u}_k) = L(\tilde{u}_k) - L(u_k) - L_c y_{t,1}. \tag{2.4.8}$$



Obr. 2.5: Ukázka nejpravděpodobnější cesty a jednotlivých konkurenčních cest.

Tato konečná informace poté přichází na vstup druhého dekodéru jako vstupní apriorní hodnota a pokračuje tak proces iterativního dekódování (viz obr. 2.4).
Apriorní hodnota se od výsledného měkkého výstupu odečítá proto, aby se tato informace nedostala zpět do dekodéru, ve kterém byla vyprodukována. Podobně systematická část, přijatá z přenosového kanálu, se odečítá z důvodu odstranění této společné informace v SOVA dekodérech. Konečné výsledky pro náš příklad jsou přehledně uvedeny v tabulce 2.6.

k	0	1	2	3	4		5		6		7		8		9		10	
1	**0**	-4	-8	-8	-8	0	-4	12	16	8	16	8	16	16	**12**	**28**	**32**	**16**
2				0	8	0	**12**	**4**	8	8	8	16	**24**	**8**	20	12		
3			**4**	4	**-12**	**12**	0	8	4	12	**20**	**4**	12	20				
4				4	-4	4	8	0	4	12	12	12	12	12				
5		**4**	0	-8	-8	0	4	4	**16**	**16**	16	8						
6				**8**	4	8	12	4	8	8	16	8						
7			4	-4	-4	4	0	8	4	12	12	12						
8				4	-4	4	0	8	12	4	12	12						

Tab. 2.5: Tabulka hodnot akumulované metriky pro jednotlivé uzly.

krok	$\tilde{u}_k$	Δ_k^s	update sekvence	úseky ovliv. $\tilde{u}_k$	min Δ_k^s	$L(\tilde{u}_k)$	$L(u_k)$	$L_c y_{t,1}$	$L_e(\tilde{u}_k)$
1	1	–	–	4,6,9	0	0	0	4	-4
2	-1	–	–	5,6,7,9	0	0	0	4	-4
3	-1	–	–	4,7	16	-16	0	-4	-12
4	1	24	1101	4,5	8	8	0	4	4
5	1	8	11010	5,8,10	8	8	0	4	4
6	-1	0	100011	6,1	0	0	0	-4	4
7	-1	16	1000110	7,8	16	-16	0	-4	-12
8	1	16	11010000	8,9	16	16	0	4	12
9	1	16	110000011	9	16	16	0	4	12
10	-1	16	1000110000	10	16	-16	0	-4	-12

Tab. 2.6: Tabulka jednotlivých výpočtů u SOVA algoritmu.

2.5 Maximum a-posteriori (MAP) algoritmus

MAP algoritmus patří, spolu se SOVA algoritmem, mezi nejčastěji používané algoritmy pro dekódování turbokódů a rovněž poskytuje měkký výstup pro jednotlivé dekódované bity. MAP algoritmus prochází během procesu dekódování dvakrát příslušným mřížovým grafem. Jeden průchod skrz mřížový graf je v dopředném a druhý ve zpětném směru (viz obrázek 2.6). Po tomto dvojitém průchodu jsou, pro všechny uzly mřížového grafu, získány stavové pravděpodobnosti $\alpha_k(s)$ a $\beta_k(s)$. Detailní matematické odvození následujících vzorců je uvedeno v literatuře [9, 10, 11].

2.5.1 Průchod v dopředném směru

Jako první se prochází mřížovým grafem v dopředném směru a pro každý uzel (stav) je současná stavová pravděpodobnost $\alpha_k(s)$ vypočtena jako součin odpovídající stavové pravděpodobnosti v předchozím uzlu $\alpha_{k-1}(s')$ a pravděpodobnosti přechodu $\gamma_k(s', s)$.

Pravděpodobnost přechodu je určena přijatým párem bitů (systematický bit a jeho paritní bit) v čase *k*. Pro obecný výpočet $\alpha_k(s)$ platí následující vztah

$$\alpha_k(s) = \sum_{s'} \alpha_{k-1}(s')\gamma_k(s', s), \tag{2.5.1}$$

kde s je současný stav, s' je předchozí stav a počáteční podmínka je

$$\alpha_0(s) = \begin{Bmatrix} 1...pokud...s = 1 \\ 0...pokud...s \neq 1 \end{Bmatrix}. \tag{2.5.2}$$

Tato počáteční podmínka znamená, že mřížový graf začíná vždy ve stavu 1 (vynulované paměťové registry).

2.5.2 Průchod ve zpětném směru

Následuje průchod mřížovým grafem ve zpětném směru. Tentokrát dochází k výpočtu stavové pravděpodobnosti $\beta_k(s)$, pro niž platí vztah

$$\beta_k(s) = \sum_{s'} \beta_{k+1}(s')\gamma_{k+1}(s, s'), \tag{2.5.3}$$

kde jednotlivé symboly mají stejný význam jako v předchozím případě, pouze β značí zpětnou stavovou pravděpodobnost. Počáteční podmínka je v tomto případě

$$\beta_N(s) = \begin{Bmatrix} 1...pokud...s = 1 \\ 0...pokud...s \neq 1 \end{Bmatrix}. \tag{2.5.4}$$

Z této podmínky vyplývá, že mřížový graf končí opět ve stavu 1.

2.5.3 Výpočet pravděpodobnosti jednotlivých přechodů

Pravděpodobnost přechodů mezi jednotlivými stavy je dána vztahem

$$\gamma_k(s', s) = A_k \pi_k \exp\left[\frac{1}{\sigma^2}\left(u_k y_{k,1} + x_k y_{k,2}\right)\right], \tag{2.5.5}$$

kde u_k a x_k je vysílaná dvojice bitů (systematický a jeho paritní bit), $y_{k,1}$ a $y_{k,2}$ je přijatá dvojice bitů z přenosového kanálu korespondující s vyslanou dvojicí bitů. V koeficientu A_k jsou zahrnuty jednotlivé diferenciály $y_{k,1}$ a $y_{k,2}$ (podrobněji popsáno v literatuře [9]). Koeficient π_k označuje apriorní pravděpodobnost (v případě první iterace je nastavena na hodnotu 0,5) a konečně parametr σ^2 popisuje (v našem případě) vlastnosti AWGN kanálu.

2.5.4 Výpočet měkkého výstupu pro dekódované bity

Nakonec jsou dopředné a zpětné pravděpodobnosti v každém úseku mřížového grafu použity k určení měkkého odhadu, zda hodnota vysílaného datového bitu u_k byla 1 nebo 0. Pro výpočet tohoto měkkého odhadu platí vztah

$$L(\tilde{u}_k) = \log \frac{\sum_{(s',s) \Rightarrow u_k = +1} \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)}{\sum_{(s',s) \Rightarrow u_k = -1} \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)}. \tag{2.5.6}$$

Výraz $\sum_{(s',s)\Rightarrow u_k=+1}(...)$ v předchozím vztahu znamená, že se v čitateli počítá s těmi přechody, které jsou vyvolány vstupním bitem $u_k = +1$ a naopak ve jmenovateli s těmi, jež jsou vyvolány vstupním bitem $u_k = -1$. I v případě MAP algoritmu platí, že můžeme měkký výstup rozložit na tři rozdílné části

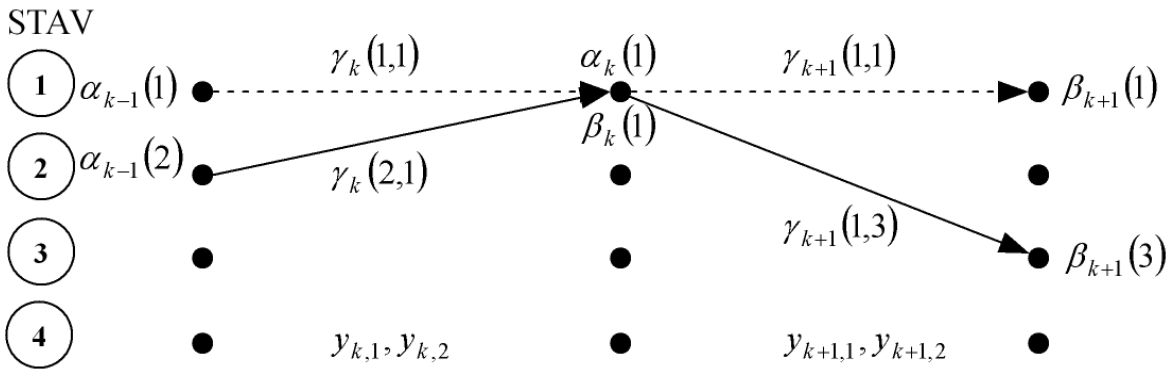
$$L(\tilde{u}_k) = L(u_k) + L_c y_{k,1} + L_e(\tilde{u}_k), \tag{2.5.7}$$

kde význam jednotlivých částí je stejný, jako v případě SOVA algoritmu (vztah 2.4.4). Výsledná informace MAP dekodéru v časovém okamžiku *k* tedy bude

$$L_e(\tilde{u}_k) = L(\tilde{u}_k) - L(u_k) - L_c y_{k,1}. \tag{2.5.8}$$

Tato konečná informace je následně přivedena na druhý MAP dekodér jako vstupní apriorní informace a pokračuje tak proces iterativního dekódování.

Z výše uvedeného je patrné, že je nejprve nutno vypočítat pravděpodobnosti všech možných přechodů $\gamma_k(s',s)$ v mřížovém grafu. Poté následuje výpočet všech dopředných stavových pravděpodobností $\alpha_k(s)$ a zpětných stavových pravděpodobností $\beta_k(s)$. Teprve na základě těchto informací jsme schopni vypočítat jednotlivé měkké výstupy $L(\tilde{u}_k)$. MAP algoritmus obvykle nebývá v iterativních dekodérech turbokódů přímo implementován, a to z důvodu velké výpočetní náročnosti tohoto algoritmu. Nejčastěji se implementuje modifikovaný MAP algoritmus, označovaný jako Max-Log-MAP algoritmus, který výrazným způsobem redukuje výpočetní náročnost MAP algoritmu [10].



$$\alpha_k(1) = \alpha_{k-1}(1)\gamma_k(1,1) + \alpha_{k-1}(2)\gamma_k(2,1)$$
$$\beta_k(1) = \beta_{k+1}(1)\gamma_{k+1}(1,1) + \beta_{k+1}(3)\gamma_{k+1}(1,3)$$

Obr. 2.6: Ukázka výpočtu stavových pravděpodobností $\alpha_k(1)$ a $\beta_k(1)$.

3 Využití turbokódů ve sdělovacích systémech

3.1 Nejznámější aplikace turbokódů ve sdělovacích systémech

V současných sdělovacích systémech jsou nejčastěji využívány turbokódy, jejichž základní parametry jsou uvedeny v následující tabulce [7].

Aplikace	Turbokód	Polynomy	Kódové rychlosti
UMTS (3G)	binární, 8-stavový	13, 15	1/3
cdma2000 (3G)	binární, 8-stavový	13, 15, 17	1/2, 1/3, 1/4
DVB-RCS (return channel)	duo-binární, 8-stavový	15, 13, 11	1/3 až 6/7
DVB-RCT (return channel)	duo-binární, 8-stavový	15, 13	1/2, 3/4
CCSDS (deep space)	binární, 16-stavový	23, 33, 25, 37	1/2, 1/3, 1/4, 1/6
INMARSAT (M4)	binární, 16-stavový	23, 35	1/2
EUTELSAT (Skyplex)	duo-binární, 8-stavový	15, 13	4/5, 6/7
IEEE 802.16 (WiMAX)	duo-binární, 8-stavový	15, 13	1/2 až 7/8

Tab. 3.1: Nejznámější aplikace turbokódů ve sdělovacích systémech.

Následuje podrobnější popis jednotlivých turbokódů z tabulky 3.1.

3.2 Turbokódy pro systémy 3G

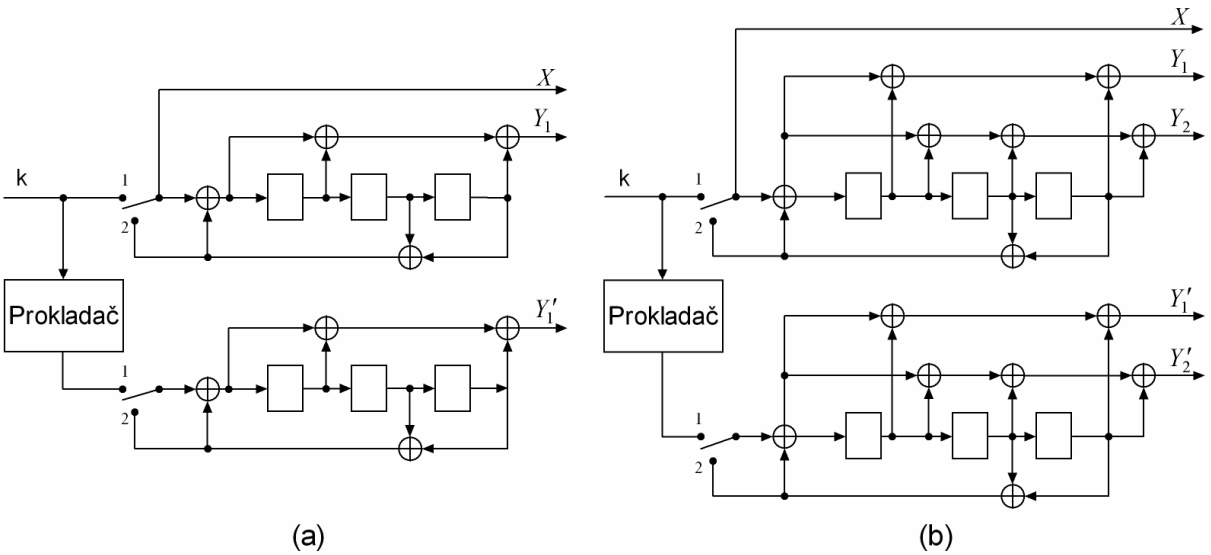
3.2.1 UMTS turbokód

UMTS (Universal Mobile Telecommunications System) je jedním ze dvou nejvíce rozšířených celulárních systémů 3G. Je standardizován organizací 3GPP (Third Generation Partnership Project) a jeho specifikace je volně dostupná na webových stránkách 3GPP http://www.3gpp.org. Pro korekční zabezpečení přenášených dat může UMTS využít konvoluční kódy nebo turbokódy (záleží na dané aplikaci a dostupné technologii).

Turbokodér používaný pro UMTS se skládá ze dvou identických RSC kodérů s délkou kódového ohraničení $K = 4$ (obr. 3.1). Vytvářecí polynomy v osmičkové soustavě mají hodnoty 13 (rekurzivní) a 15 (paritní). Výstup UMTS turbokodéru je sériová kombinace systematických bitů $\{X_k\}$, paritního výstupu prvního RSC kodéru $\{Y_{1,k}\}$ a paritního výstupu druhého RSC kodéru $\{Y_{2,k}\}$. V čase k tedy bude na výstupu UMTS turbokodéru trojice bitů $Y_k = \{X_k, Y_{1,k}, Y_{2,k}\}$. Celková kódová rychlost je tedy $R = 1/3$.

Velikost vstupního bloku dat je volena v rozmezí 40 až 5114 bitů $(40 \le k \le 5114)$. Blok prokládání, jehož velikost závisí na délce vstupního bloku dat, změní polohu jednotlivých vstupních bitů podle předepsaného algoritmu. Tento algoritmus je dosti komplikovaný a jeho přesnou podobu lze najít v literatuře [13]. Při průchodu vstupního bloku dat oběma RSC kodéry je spínač v poloze 1. Po průchodu vstupních dat dojde k přepnutí spínače do polohy 2 a následně posuvný registr provede další tři kroky, čímž dojde k přidání

tří ukončovacích bitů za vstupní datovou posloupnost. Tím zajistíme vynulování jednotlivých paměťových buněk RSC kodéru. Poté se spínač přepne opět do polohy 1 a na vstup RSC kodérů přichází další vstupní blok dat.



Obr. 3.1: Ukázka zapojení (a) UMTS turbokodéru, (b) cdma2000 turbokodéru.

3.2.2 cdma2000 turbokód

Druhý nejvíce rozšířený celulární standard 3G je cdma2000, který je standardizován organizací 3GPP2 (viz dostupné webové stránky http://www.3gpp2.org). Stejně jako UMTS, i cdma2000 používá pro korekční zabezpečení konvoluční kódy nebo turbokódy. Oba systémy se nejvíce liší v prokládacím algoritmu, rozsahu povolených velikostí bloků vstupních dat a informační rychlostí RSC kodérů.

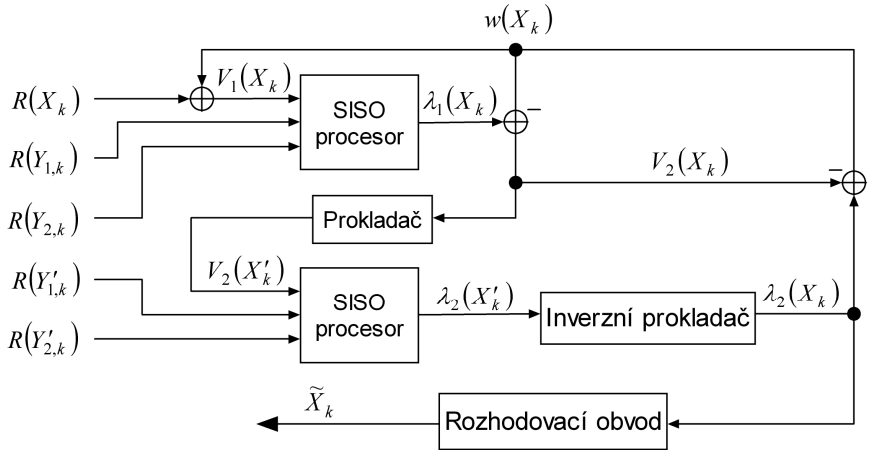
Vytvářecí polynomy mají hodnoty 13 (systematický), 15 (první paritní) a 17 (druhý paritní). Velikost vstupního bloku dat pro cdma2000 turbokód může být volena pouze z hodnot: 378, 570, 762, 1146, 1530, 2398, 3066, 4602, 6138, 9210, 12282, 20730 bitů. Rozdíl je také ve způsobu prokládání [13].

Zapojení cdma2000 turbokodéru je vidět na obrázku 3.1. RSC kodér má v tomto případě 3 výstupy (jeden systematický a dva paritní) a celková kódová rychlost pro cdma2000 turbokód je tedy $R = 1/5$. Pro mnoho aplikací je tato nízká rychlost nežádoucí, proto cdma2000 používá děrování pro zvýšení celkové kódové rychlosti. Vynecháním některých paritních bitů je dosaženo zvýšení kódové rychlosti na hodnoty $R = \left\{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}\right\}$.

Pokud budou přenášeny pouze první paritní výstupy u obou RSC kodérů, bude mít cdma2000 turbokodér stejnou strukturu jako v případě UMTS. Dále se ale budou oba systémy lišit v prokládacím procesu a povolených velikostech vstupních bloků. Ukončovací bity jsou za vstupní data přidávány stejným způsobem, jako v případě UMTS.

3.2.3 Dekódování UMTS a cdma2000 turbokódů

Dekódovací architektura z obrázku 3.2 může být univerzálně použita pro oba systémy, UMTS i cdma2000. V případě UMTS budou využity pouze paritní vstupy $R(Y_{1,k})$ a $R(Y'_{1,k})$, přičemž na ostatních vstupech dekodéru budou samé nuly. U cdma2000 bude využití vstupů dekodéru záviset na použitém děrování na straně turbokodéru. Pro různé kódové rychlosti daného turbokódu budou použity odpovídající vstupy turbodekodéru [13].



Obr. 3.2: Architektura pro společné dekódování UMTS a cdma2000 turbokódu.

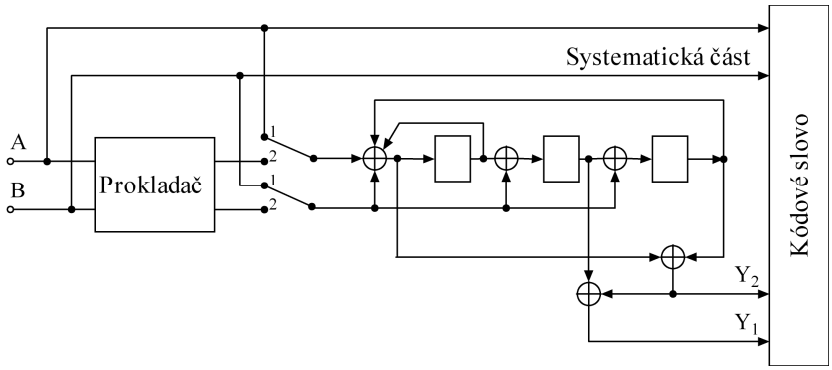
3.3 Turbokódy použité v DVB (Digital Video Broadcasting)

3.3.1 DVB–RCS (Return Channel over satellite) turbokód

Tento standard nabízí asymetrickou širokopásmovou komunikaci. Po dopředném kanálu se rychlost pohybuje v jednotkách až desítkách Mb/s a po zpětném kanálu pak až 2 Mb/s. V dopředném kanále se používá zapouzdření IP do formátu MPEG-2. Zpětný kanál (RSC) pracuje na principu multifrekvenčního časového multiplexu (umožňuje vysílat každý paket na jiné frekvenci). V tomto zpětném kanálu se provádí zapouzdření IP do ATM buněk nebo opět do formátu MPEG-2 [12].

Je použit duo-binární, 8-stavový CRSC (circular RSC) kodér s vytvářecími polynomy 15 (systematický), 13 (první paritní) a 11 (druhý paritní). Podporované kódové rychlosti jsou $R = \left\{\frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{6}{7}\right\}$. Druhý paritní výstup je využíván pouze pro kódové rychlosti menší než 1/2 (tzn. 1/3 a 2/5).

Podporované velikosti vstupních datových bloků jsou: 12, 16, 53, 55, 57, 106, 108, 110, 188, 212, 214 a 216 bytů, což zahrnuje i ATM (53 bytů) a formát MPEG-2 (188 bytů). V prodeji je dostupné jednoduché čipové řešení označované jako TC1000 turbo kodér/dekodér. Schéma kodéru pro DVB-RCS turbokód je vidět na obrázku 3.3.



Obr. 3.3: Schéma kodéru pro DVB-RCS turbokód.

3.3.2 DVB–RCT (Return Channel over Terrestrial) turbokód

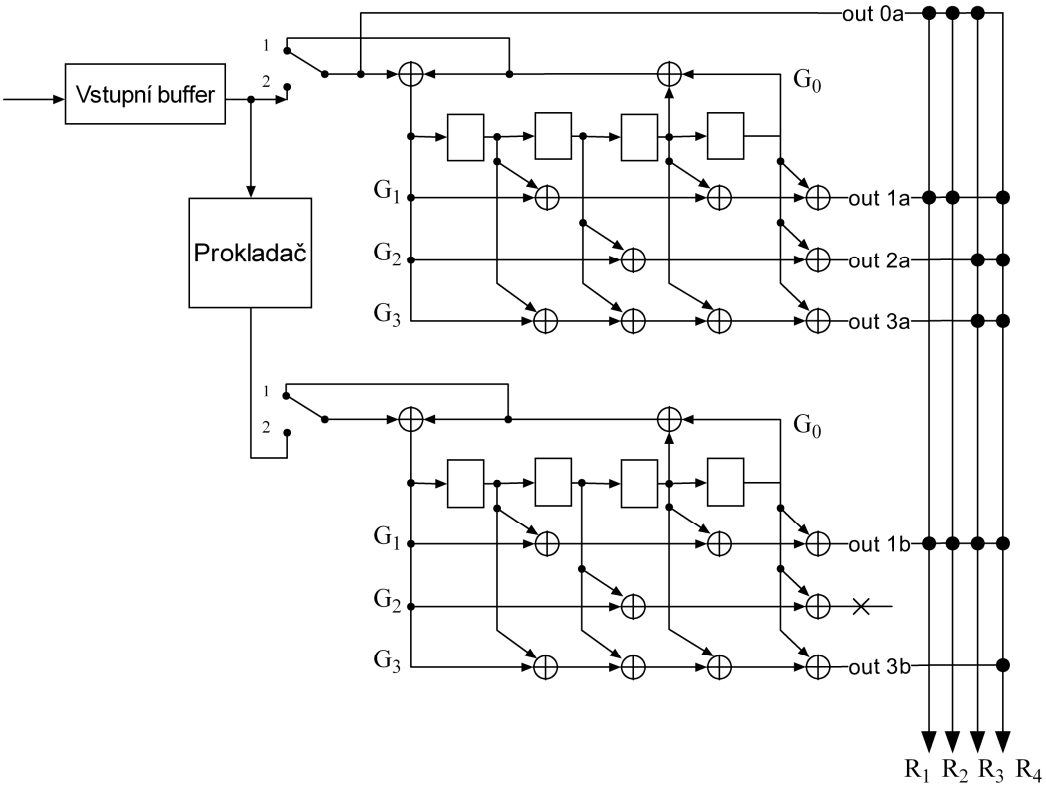
Jedná se opět o zavedení zpětného kanálu pro interaktivní, ale tentokrát pro pozemní digitální televizi [13]. Standard DVB-RCT využívá stejný turbokód jako DVB-RCS s následujícími rozdíly:

1. Je podporováno pouze 5 velikostí vstupního bloku dat, a to v rozsahu od 18 do 81 bytů. Je vidět, že jsou voleny poměrně malé velikosti vstupních bloků, narozdíl od standardu DVB-RCS.
2. Jsou podporovány pouze dvě kódové rychlosti $R = \left\{\frac{1}{2}, \frac{3}{4}\right\}$.

3.4 CCSDS turbokód

Zhruba před 20 lety byl v oblasti komunikačních protokolů určených pro kosmické výpravy založen Consultative Commitee for Space Data Systems (CCSDS). Jeho hlavním úkolem bylo stanovení mezinárodně platných standardů pro komunikaci mezi vzdálenými kosmickými sondami a pozemskými řídícími centry.

Pro tyto účely byl navržen turbokód, který je tvořen RSC kodérem s délkou kódového ohraničení $K = 5$ a vytvářecími polynomy 23 (rekurzivní), 33 (první paritní), 25 (druhý paritní), 37 (třetí paritní). Jsou definovány 4 velikosti vstupních bloků dat: 223, 446, 892, 1115 bytů (je definována i pátá velikost 2048 bytů, ale není ještě specifikována). CCSDS doporučení podporuje kódové rychlosti $R = \left\{\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{6}\right\}$, které na obrázku 3.4 odpovídají označení $\{R_1, R_2, R_3, R_4\}$. Na tomto obrázku je rovněž přehledně vidět využití jednotlivých paritních výstupů pro dané kódové rychlosti.



Obr. 3.4: Schéma kodéru pro CCSDS turbokód.

3.4 Další turbokódy používané ve sdělovacích systémech současnosti

3.4.1 INMARSAT (M4)

INMARSAT je družicový komunikační systém, který používá 4 družice na geostacionárních drahách. Původní rozsah jeho služeb byl zaměřen na oblast námořní komunikace, ale později se pole jeho působnosti rozšířilo a nyní pokrývá i oblast pozemní a letecké komunikace. Definuje několik různých standardů, které se odlišují hlavně v nabízených službách. Jedním z těchto standardů je INMARSAT (M4), který využívá následující turbokód.

Jedná se o binární, 16-stavový turbokód, jehož vytvářecí mnohočleny jsou 23, 25 a použitá kódová rychlost je $R = 1/2$. Ukázka zapojení tohoto turbokódu je na obr. 3.5.

3.4.2 EUTELSAT (Skyplex)

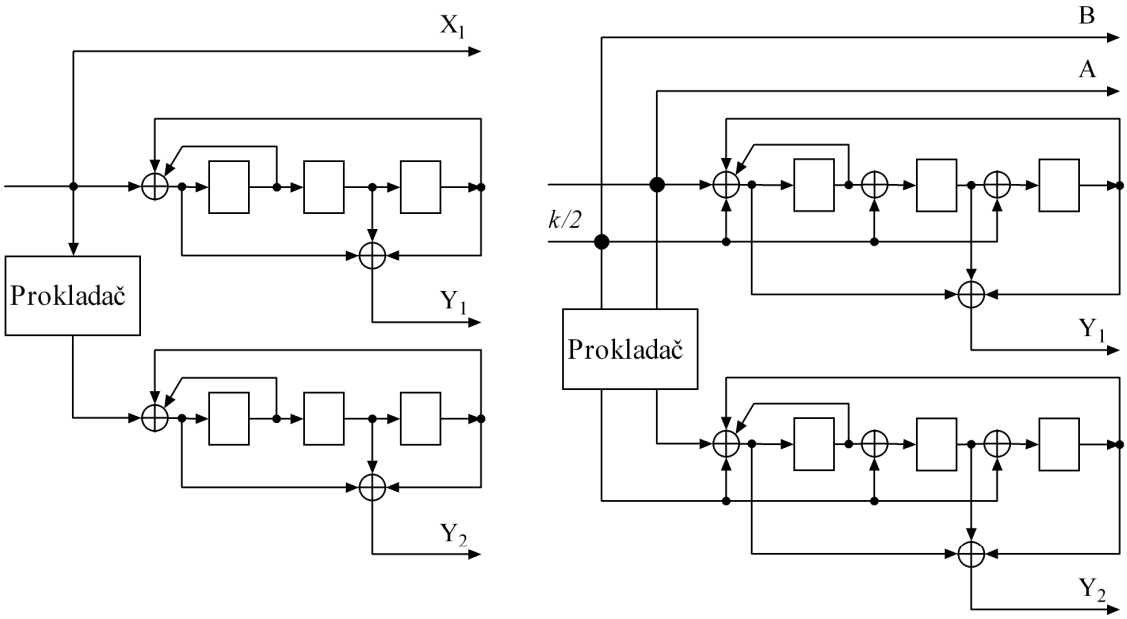
Eutelsat S.A. je francouzská společnost, která se stala hlavním satelitním operátorem v Evropě a navíc patří mezi tři nejlepší poskytovatele pevných satelitních služeb na světě. Umožňuje vysokorychlostní přístup k informacím miliónům domácností po celém světě. Využívá 23 družic (19 má ve svém vlastnictví). Ty tvoří dostatečnou kapacitu přenosových linek pro operátory, kteří poskytují jejich zákazníkům televizní a rádiové služby.

EUTELSAT (Skyplex) využívá duo-binární, 8-stavový turbokód s vytvářecími mnohočleny 15,13. Podporované kódové rychlosti jsou $R = \left\{\frac{4}{5}, \frac{6}{7}\right\}$. Ukázka zapojení je opět uvedena na obr. 3.5.

3.4.3 IEEE 802.16 (WiMAX)

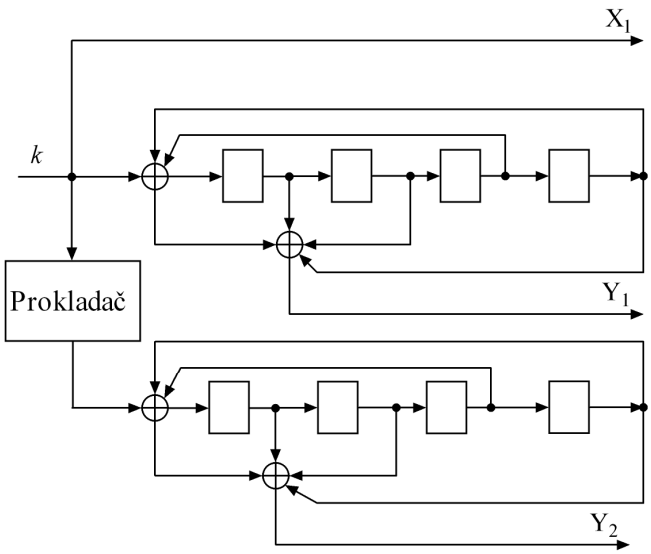
WiMAX (Worldwide Interoperability for Microwave Access) lze zařadit do skupiny bezdrátových způsobů připojení. Jeho počátky se datují do let 2000 až 2003, kdy vznikl nový standard IEEE 802.16. Oproti staršímu standardu WiFi, určenému především pro vnitřní sítě, je WiMAX využíván výhradně pro venkovní prostředí, kde může dosahovat vyšších přenosových rychlostí (teoreticky až 75 Mb/s). Dále tento standard nabízí vyšší dosah (teoreticky až 50 km) a možnost spojení i bez přímé viditelnosti (teoreticky do 3 km). Největší výhodou je ovšem bezesporu podpora řízení kvality služeb (QoS - Quallity of Service).

Tento standard využívá stejný turbokód jako výše popsaný EUTELSAT, s tím rozdílem, že je podporováno více kódových rychlostí v rozmezí $R = \left\{\frac{1}{2} až \frac{7}{8}\right\}$.



(a) binární, 8-stavový **(b)** duo-binární, 8-stavový

Nejčastěji používané polynomy 15, 13 nebo (13, 15).



(c) binární, 16-stavový

Nejčastěji používané polynomy 23, 35 nebo (31, 27).

Obr. 3.5: Ukázka dalších, často používaných turbokódů z tabulky 3.1.

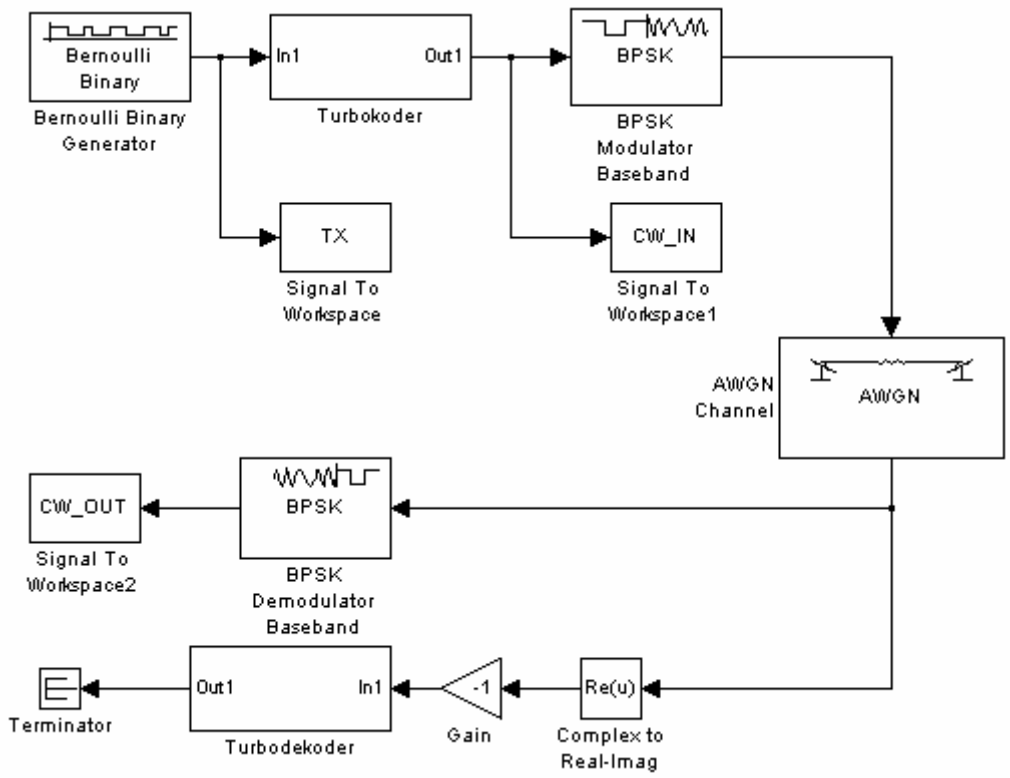
4 Parametry a popis simulace

Tato kapitola se věnuje detailnějšímu popisu jednotlivých modelů a jejich bloků použitých k simulaci zabezpečovacích schopností turbokódů v programu SIMULINK. U těchto schémat jsem nepovažoval za nutné překládat názvy použitých bloků do češtiny, proto jsou na obrázcích uvedeny originální anglické popisky. Dále budou popsány důležité části obslužného skriptu v programu MATLAB a jeho možné modifikace. Výsledky všech simulací budou prezentovány až v následující kapitole.

4.1 Základní popis modelu použitého při simulaci

Obrázek 4.1 ukazuje základní schéma zapojení použité pro simulaci korekčních schopností turbokódů (bez použití děrování). Vstupní datová posloupnost se generuje v bloku Bernoulli Binary Generátor a je dále přiváděna na vstup turbokodéru (vnitřní struktura bloku turbokodéru i turbodekodéru bude popsána později). Na zakódovanou posloupnost je poté aplikována BPSK modulace v základním pásmu a takto upravená data následně vstupují do přenosového kanálu. Pro tuto simulaci jsem použil model AWGN přenosového kanálu, takže se ke vstupnímu signálu přičítá bílý Gausovský šum. Na přijímací straně není potřeba přijatá data demodulovat, protože blok turbodekodéru může akceptovat i nekvantizovaný vstup. Stačí tedy z přeneseného signálu odebírat reálnou složku a takto získané hodnoty poté mapovat do správného rozsahu (v našem případě stačí dané hodnoty invertovat) a přivádět dále na vstup turbodekodéru.

Před vstupem do turbokodéru je vstupní datová posloupnost ukládána do pracovního prostředí programu MATLAB, a to do proměnné označené jako TX. Hodnoty této proměnné jsou na konci simulace porovnány s hodnotami jednotlivých bitů na výstupu turbodekodéru, což umožňuje určit chybovost přenosu. K podobnému účelu slouží i další proměnné, CW_IN a CW_OUT.

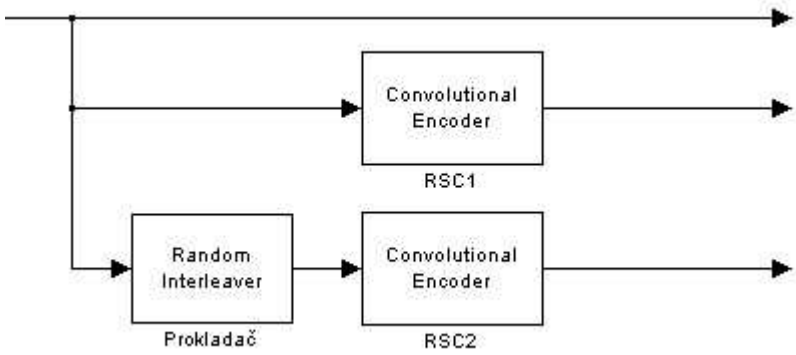


Obr. 4.1: Základní model použitý při simulaci (bez děrování).

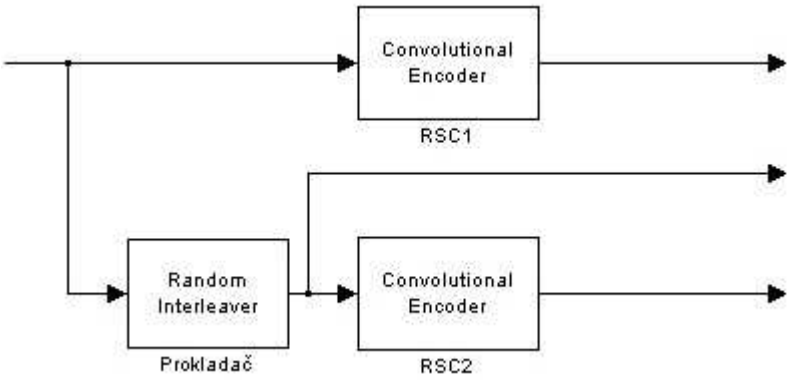
4.2 Problematika ukončovacích bitů

Úkolem přídavných bitů je uvést kodér do takového stavu, kdy po průchodu takto upraveného vstupního bloku dat kodérem budou všechny jeho paměťové buňky vynulovány. Těchto *m* přidaných bitů není možné jednoduše předpovídat, protože závisí na konkrétním stavu kodéru. V kapitole 1.4 byl popsán způsob, jakým se přidávají ukončovací bity ke vstupnímu bloku dat v případě RSC kodérů.

Výstup turbokodéru je ve většině případů tvořen třemi bitovými toky, jedním systematickým a dvěma zabezpečovacími. Přenosovým kanálem se tedy přenáší systematická posloupnost (i s jejími ukončovacími bity) pouze jednoho ze dvou RSC kodérů. Na následujících obrázcích jsou vidět dva možné způsoby realizace turbokodéru se dvěma RSC kodéry.



Obr. 4.2: Ukončovací bity systematické posloupnosti jsou odvozeny v kodéru RSC1.



Obr. 4.3: Ukončovací bity systematické posloupnosti jsou odvozeny v kodéru RSC2.

V odborných publikacích se nejčastěji uvádí zapojení turbokodéru podle obrázku 4.2, kde je sada *m* ukončovacích bitů systematické posloupnosti odvozena v prvním RSC kodéru. Nicméně v těchto publikacích je také často prezentováno zapojení turbodekodéru podle obrázku 2.4, kdy se po skončení daného počtu dekódovacích kroků rozhoduje o konečné podobě dekódovaných bitů na základě měkkého výstupu druhého RSC kodéru. Při praktické realizaci tohoto turbodekodéru se přenesená systematická posloupnost přivádí na vstup druhého RSC dekodéru přes blok prokladače. Tímto prokladačem prochází ovšem systematická část bez ukončovacích bitů a za takto proloženou posloupnost se poté přidají ukončovací bity získané z prvního RSC kodéru. Tento způsob dekódování není vhodný, protože jednotlivé ukončovací bity tohoto systematického bitového toku souvisí s prvním

RSC kodérem, což v důsledku způsobí, že konečná dekódovaná cesta mřížovým grafem nebude ve většině případů rovna nejvíce pravděpodobné cestě. Celková korekční schopnost turbokódů se v tomto případě podstatným způsobem zhorší. Tuto vlastnost jsem měl možnost si osobně ověřit při tvorbě těchto simulací, kdy jsem zpočátku vycházel z chybných informací uvedených v literatuře [8, 11] a takto získané výsledky neodpovídaly teoretickým předpokladům.

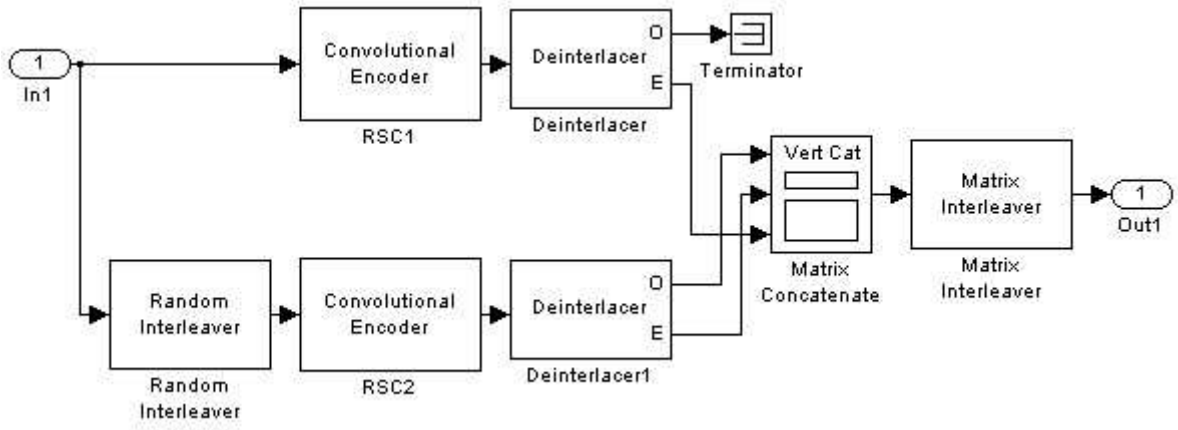
Při realizaci turbokodéru podle obrázku 4.2 je tedy nutné, aby rozhodování o konečné podobě dekódovaných bitů bylo prováděno v prvním RSC dekodéru. Pokud použijeme zapojení turbokodéru z obrázku 4.3, bude možné zachovat strukturu turbodekodéru podle obrázku 2.4 (rozhodování v druhém RSC dekodéru) s tím, že bude odstraněn prokladač před vstupem systematické části do druhého RSC dekodéru a naopak přidán inverzní prokladač před vstup systematické části do prvního RSC dekodéru.

4.3 Vnitřní struktura turbokodéru

Vnitřní strukturu bloku turbokodéru použitého při simulaci ukazuje následující obrázek. Vstupní datová sekvence se přivádí do prvního RSC kodéru přímo a do druhého RSC kodéru přes blok prokládání. Při základním nastavení je použit pseudonáhodný prokladač, výjimku tvoří simulace, která bude mít za úkol testovat vliv prokladače na korekční schopnosti turbokódů. V tomto případě bude použit i klasický blokový prokladač.

Po zakódování vstupních dat v RSC kodérech dochází k rozdělení jednotlivých bitových toků v bloku Deinterlacer, a to z toho důvodu, že výstup RSC kodéru v prostředí programu SIMULINK je tvořen posloupností bitů, kde na liché pozici jsou umístěny systematické bity a na sudé pozici bity zabezpečovací. Na výstupu tohoto bloku jsou tedy k dispozici dva bitové toky (systematický a zabezpečovací), se kterými je možné dále samostatně pracovat.

Z hlediska problematiky ukončovacích bitů bude při všech simulacích použito zapojení podle obrázku 4.3, proto systematická část prvního RSC kodéru na obrázku 4.4 není dále přenášena. Vlastnosti zbývajících dvou bloků z obrázku 4.4 není potřeba detailně rozebírat. Jejich úkolem je sloučit jednotlivé bitové toky do jednoho takovým způsobem, abychom na výstupu turbokodéru dostali zakódovanou sekvenci bitů ve tvaru popsaném v první kapitole. V našem případě je pro každý vstupní nezabezpečený bit tento požadovaný tvar určen trojicí bitů (systematického bitu z druhého RSC kodéru, zabezpečovacím bitem generovaným ve druhém RSC kodéru a zabezpečovacím bitem z prvního RSC kodéru).

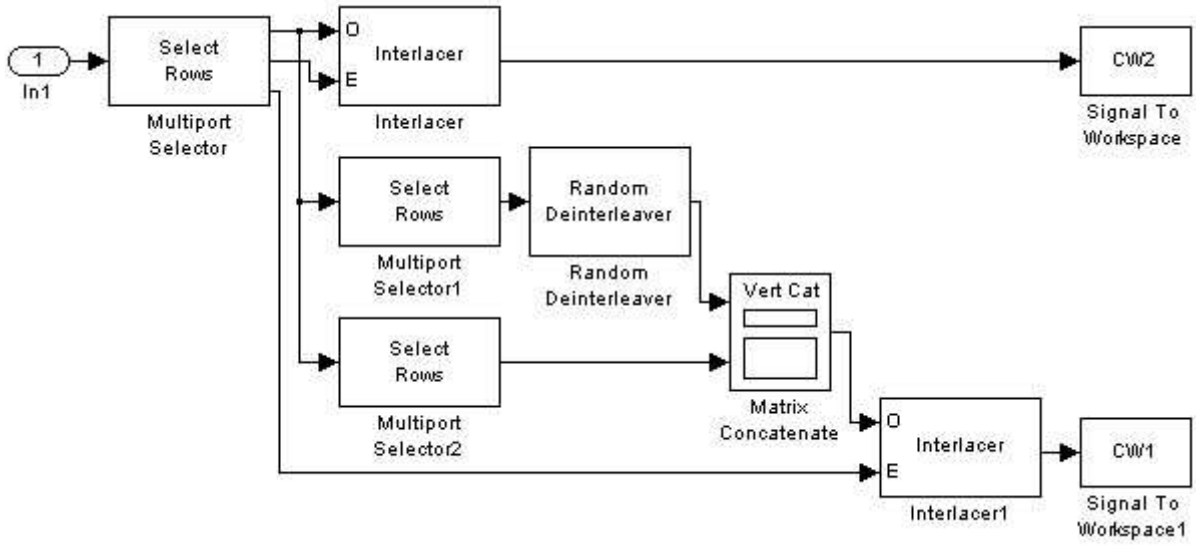


Obr. 4.4: Schéma zapojení turbokodéru použitého při simulaci.

4.4 Vnitřní struktura turbodekodéru

Na vstup turbodekodéru přichází datový tok přijatý z přenosového kanálu. Tento datový tok je v prostředí programu SIMULINK reprezentován formou sloupcového vektoru. Blok Select Rows na vstupu turbodekodéru je nastaven tak, aby tento sloupcový vektor rozdělil na tři samostatné datové toky. Jeho jednotlivé výstupy tedy tvoří: systematická část náležející druhému RSC kodéru, zabezpečovací posloupnost druhého RSC kodéru a zabezpečovací část prvního RSC kodéru. Systematická a zabezpečovací část z druhého RSC kodéru jsou pomocí bloku Interlacer sloučeny do jednoho toku tak, že na liché pozici jsou umístěny hodnoty systematické části a na sudé pozici hodnoty ze zabezpečovací části. Takto upravená posloupnost se nakonec uloží do pracovního prostředí programu MATLAB, do proměnné s názvem CW2.

Dále obrázek 4.5 ukazuje, že z přenesené systematické posloupnosti je opět pomocí bloku Select Rows oddělena část odpovídající systematické části druhého RSC kodéru bez *m* ukončovacích bitů, která je přivedena do inverzního pseudonáhodného prokladače. Zbývající blok Select Rows se používá k oddělení *m* ukončovacích bitů z přenesené systematické části, jež jsou následně připojeny na konec datové posloupnosti získané na výstupu inverzního pseudonáhodného prokladače. Pomocí bloku Interlacer se nakonec, stejně jako v případě CW2, získá druhá proměnná nazvaná CW1 (hodnoty na liché pozici tentokrát náleží systematické části prvního RSC kodéru a hodnoty na sudé pozici odpovídají zabezpečovací části prvního RSC kodéru).



Obr. 4.5: Schéma zapojení turbodekodéru použité při simulaci.

Proměnné CW1 a CW2 jsou uloženy do pracovního prostředí programu MATLAB, kde poté dochází k jejich dalšímu zpracování pomocí dekódovacího SOVA algoritmu. Výstupem dekódovacího algoritmu u jednotlivých RSC dekodérů je:

1) Měkký výstup $L(\tilde{u}_k)$.
2) Apriorní informace $L_e(\tilde{u}_k)$ určená pro následující RSC dekodér.
3) Tvrdý výstup získaný následujícím způsobem:
   - Pokud je hodnota měkkého výstupu menší nebo rovna nule, bude výsledný dekódovaný bit mít hodnotu 0.
   - Naopak, pokud bude hodnota měkkého výstupu větší než nula, výsledný dekódovaný bit je roven 1.

Na konci každého dekódovacího kroku je tedy získán tvrdý výstup z druhého RSC dekodéru, který se porovná se vstupní datovou posloupností uloženou v proměnné TX (viz obrázek 4.1). Výsledkem tohoto porovnání bude počet chyb, které daný turbokód nebyl schopen opravit, což umožňuje určit výslednou chybovost tohoto přenosu. Podobně existuje možnost porovnat proměnné CW_IN a CW_OUT a tím zjistit, kolik bitů zakódované posloupnosti bylo chybně přeneseno AWGN kanálem.

4.5 Popis důležitých částí skriptu v programu MATLAB

V kapitole 5 bude předvedeno, že celková korekční schopnost daného turbokódu závisí na mnoha faktorech, které je možné zohlednit při návrhu turbokódu. Vliv každého z těchto faktorů byl testován pomocí samostatného skriptu v programu MATLAB. Tyto skripty jsou součástí přiloženého CD a není tedy nutné se zabývat jejich detailním rozborem. V další části textu se chci zaměřit pouze na stěžejní části těchto skriptů, které se v těchto skriptech opakují a jsou důležité z hlediska správného pochopení dané problematiky nebo z hlediska případných modifikací.

4.5.1 Inicializace proměnných

```
%----INICIALIZACE PROMĚNNÝCH----
K = 4;      %Délka kódového ohraničení RSC kodéru.
GP1 = 15;   %Generující polynom 1 osmičkové soustavě.
GP2 = 17;   %Generující polynom 2 osmičkové soustavě.
ZS = 15;    %Generující polynom odpovídající zpětnovazební smyčce.

RandINTRLV = 64978;     %Inicializační hodnoty pro generátor náhodných
RandBG = 42591;         %čísel v bloku pseudonáhodného prokladače,
RandAWGN = 3574;        %Bernoulliho generátoru a AWGN kanálu.

Delka = 1000;               %Délka jednoho bloku dat v bitech.
Delka_m = Delka + (K-1);    %Délka bloku + m ukončovacích bitů.

PK = 8;     %Počet dekódovacích kroků.
Rc = 1/3;   %Celková kódová rychlost turbokodéru.
N = 500;    %Celkový počet bloků.
W = 2;      %Váhový koeficient pro dělení apriorních informací.

EbNodB = 0:+0.25:4; %Rozsah hodnot Eb/Eo v decibelech.

Struktura = poly2trellis(K, [GP1 GP2],ZS);  %Definice RSC kodéru.
```

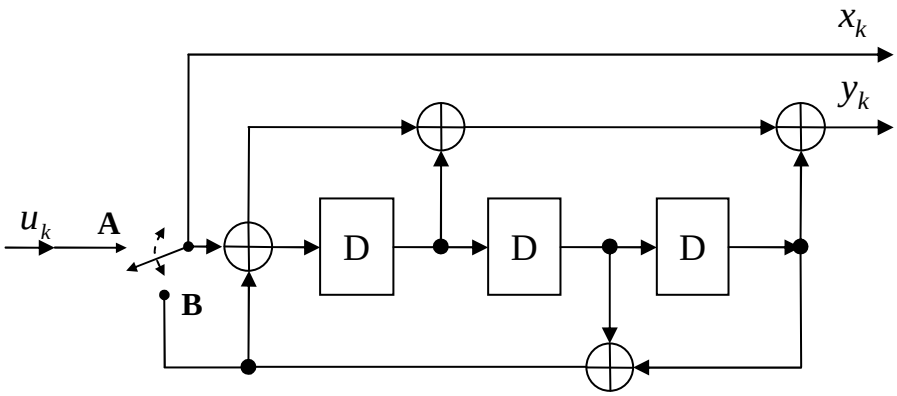
Obr. 4.6: Část skriptu – Inicializace proměnných.

Inicializační hodnoty pro generátor náhodných čísel v blocích Bernoulliho generátoru a AWGN kanálu lze zvolit náhodně. Pouze v případě pseudonáhodného prokladače a inverzního pseudonáhodného prokladače je nutné nastavit stejnou inicializační hodnotu. Ta nám zajistí, že k přeskupení jednotlivých bitů v rámci jednoho datovém bloku bude v obou blocích použita stejná náhodná permutace.

4.5.2 Nastavení bloku konvolučního kodéru

Pro nastavení bloku konvolučního kodéru je potřeba správně definovat strukturu použitého konvolučního kódu. U RSC kodérů musí platit, že generující polynom GP1 patřící

k prvnímu výstupu (v případě RSC kodéru se jedná o systematický výstup) a generující polynom představující zpětnovazební smyčku, musí mít stejnou hodnotu. Pro názornost uvádím příklad RSC kodéru z druhé kapitoly, jehož schéma zapojení lze vidět na následujícím obrázku.

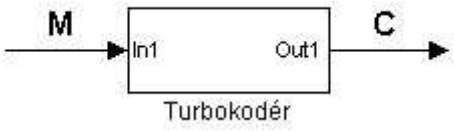


Obr. 4.7: Ukázka zapojení RSC kodéru.

První generující polynom musí odpovídat zpětnovazební smyčce, protože první výstup je systematický. Zpětnovazební generující polynom je reprezentován binárním vektorem [1011], což v osmičkové soustavě odpovídá hodnotě 13. Druhý generující polynom definuje binární vektor [1101], což v osmičkové soustavě odpovídá číslu 15. Délka kódového ohraničení tohoto RSC kodéru je $K = 4$. V programu MATLAB lze využít při generování mřížového grafu pomocnou funkci poly2trellis, která má v případě RSC kodéru tvar poly2trellis(K, [GP1 GP2], ZS) a pro náš příklad tedy dostáváme poly2trellis(4, [13 15], 13). Více informací k této problematice uvádí literatura [14].

4.5.3 Nastavení bloku AWGN kanálu

Následuje odvození vztahu mezi *SNR* a E_b / N_0 v případě použití BPSK modulace a AWGN kanálu [16].



Obr. 4.8: Ilustrační obrázek bloku turbokodéru.

Na vstup turbokodéru vstupuje k-bitová zpráva M, která vygeneruje na jeho výstupu n-bitové kódové slovo C. Celková energie obsažená ve vstupní zprávě je $M = kE_b$ a celková energie v kódovém slovu je $C = nE_s$ (E_b udává energii jednoho bitu, kdežto E_s vyjadřuje energii jednoho symbolu). Musí platit rovnost

$$M = C \Rightarrow kE_b = nE_s \,. \tag{4.5.1}$$

Tento vztah lze postupně upravovat následovně:

$$E_b = E_s / R_c \,, \tag{4.5.2}$$

$$E_b / N_0 = SNR / R_c \,, \tag{4.5.3}$$

kde R_c je celková kódová rychlost turbokodéru a N_0 udává spektrální hustotu výkonu šumu o rozměru (W/Hz).
Vztah (4.5.3) můžeme vyjádřit v decibelech jako

$$E_b / N_0 = SNR - 10\log_{10}(R_c). \tag{4.5.4}$$

V jednotlivých simulacích si rozsah hodnot E_b / N_0 volíme a pro každou z těchto hodnot se bloku AWGN kanálu předává odpovídající hodnota SNR podle vztahu

$$SNR = E_b / N_0 + 10\log_{10}(R_c). \tag{4.5.5}$$

4.5.4 Realizace iterativního dekódování

```
%----REALIZACE ITERATIVNÍHO ZPŮSOBU DEKÓDOVÁNÍ TURBOKÓDŮ----
for m = 1:PK,           %Smyčka pro daný počet dekódovacích kroků.

    %Podmínka zajišťující v případě prvního dekódovacího kola nastavení
    %nulových apriorních informací pro první dílčí dekodér. V ostatních
    %případech jsou použity apriorní informace z druhého dekodéru.
    if (m ==1)
    Apriori = zeros(1, (length(TX) +(K-1)));
    else
    Apriori = Le2;
    end

    %Dílčí dekodér 1
    CW = PCW1;      %Jako vstupní kódové slovo se použije CW1
    [hard, soft, Le] = sova (K, GP1, GP2, ZS, Lc, CW, Apriori, W);
    Le1 = Le(1:+1:Delka);   %Konečná informace Le1 bez ukončovacích bitů.
    UB = Le((Delka+1):+1:Delka_m);   %Ukončovací bity.
    Int = randintrlv(Le1, RandINTRLV);
    Apriori = [Int, UB];

    %Dílčí dekodér 2
    CW = PCW2;      %Jako vstupní kódové slovo se použije CW2
    [hard, soft, Le] = sova (K, GP1, GP2, ZS, Lc, CW, Apriori, W);
    Le2 = Le(1:+1:Delka);   %Konečná informace Le2 bez ukončovacích bitů.
    UB = Le((Delka+1):+1:Delka_m);   %Ukončovací bity.
    Deint = randdeintrlv(Le2, RandINTRLV);
    Le2 = [Deint, UB];
    Phard = hard(1:+1:Delka);   %tvrdý výstup (bez UB) druhého dekodéru.

    %Konečný tvrdý výstup získaný na konci každého dekódovacího kroku.
    Hard_OUT = randdeintrlv(Phard, RandINTRLV);

end
```

Obr. 4.9: Část skriptu – Realizace iterativního dekódování.

Funkce **sova** má 8 vstupních parametrů (K, GP1, GP2, ZS, Lc, CW, Apriori, W) a navrací hodnoty tří proměnných (hard, soft, Le). Význam parametrů K, GP1, GP2 a ZS byl vysvětlen v předchozím textu, Lc představuje spolehlivost přenosového kanálu a pro její výpočet platí vztah

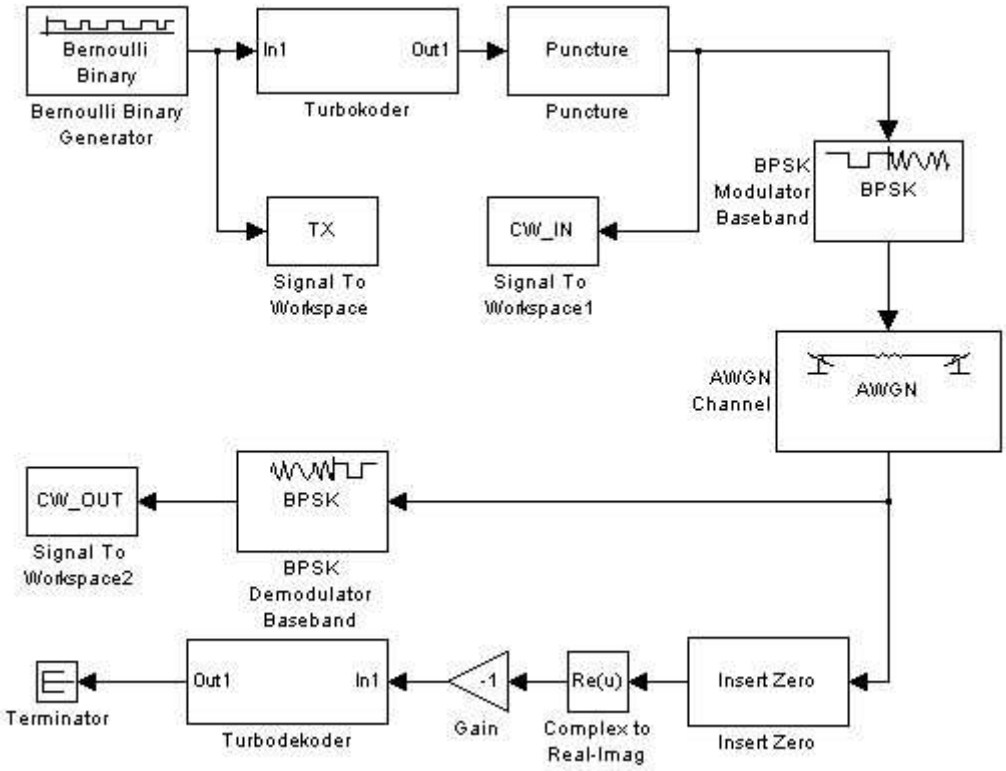
$$Lc = 4\frac{E_b}{N_0}. \tag{4.5.6}$$

CW je zakódovaná posloupnost vstupující do dílčího dekodéru. Na lichých pozicích této posloupnosti musí být systematické bity a na sudých pozicích odpovídající zabezpečovací bity. Apriori představuje apriorní informace získané z druhého dílčího dekodéru a W slouží k vážení těchto apriorních informací. Vliv parametru W na iterativní dekódovací proces bude vysvětlen v 5. kapitole. Standardně je W nastaveno na hodnotu 2.

Z konečné informace Le, získané jako výstup funkce **sova** u dílčího dekodéru 1, jsou odděleny hodnoty korespondující s ukončovacími bity. Takto upravená posloupnost Le je následně přeuspořádána na náhodnou permutaci, která se shoduje s permutací v bloku pseudonáhodného prokladače. Apriorní informace pro druhý dílčí dekodér tvoří tato přeuspořádaná posloupnost Le, na jejíž konec se přidají hodnoty korespondující s ukončovacími bity. V případě druhého dílčího dekodéru lze při získávání apriorních informací pro případné další dekódovací kolo postupovat obdobným způsobem. Na konci každého dekódovacího kola bude výstup turbodekodéru tvořen tvrdým výstupem druhého dílčího dekodéru.

Implementace SOVA algoritmu použitého v rámci této diplomové práce vychází z algoritmu, jehož zdrojový kód lze volně stáhnout na oficiálních webových stánkách k programu MATLAB [15]. Tento algoritmus byl upraven do podoby, která odpovídá popisu SOVA algoritmu ve druhé kapitole této práce. Zdrojový kód funkce **sova** je možné vyhledat na přiloženém CD.

4.6 Rozšíření základního modelu o děrování



Obr. 4.10: Rozšíření základního modelu o děrování.

V rozšířeném základním modelu dochází k děrování výstupního datového toku turbodekodéru. Struktura tohoto datového toku byla popsána v kapitole 4.1. V bloku děrování (puncture) je použit děrovací vektor [1 1 0 1 0 1], kde hodnota 0 odpovídá bitům, které nebudou dále přenášeny. Z tohoto vektoru lze vyvodit, že systematický bit se přenáší vždy a ke každému systematickému bitu je přidán pouze jeden zabezpečovací bit náležející střídavě k prvnímu nebo ke druhému RSC kodéru. Jinými slovy tedy dochází k děrování zabezpečovacích posloupností jednotlivých RSC kodérů, kdy jsou dále přenášeny pouze liché bity paritní posloupnosti prvního a sudé bity paritní posloupnosti druhého RSC kodéru. Celková informační rychlost poklesne z původní hodnoty $R = 1/3$ na $R = 1/2$. Tento pokles informační rychlosti ovšem způsobí i snížení korekčních schopností daného turbokódu. Vliv děrování na chybovost přenosu bude podrobněji popsán v následující kapitole.

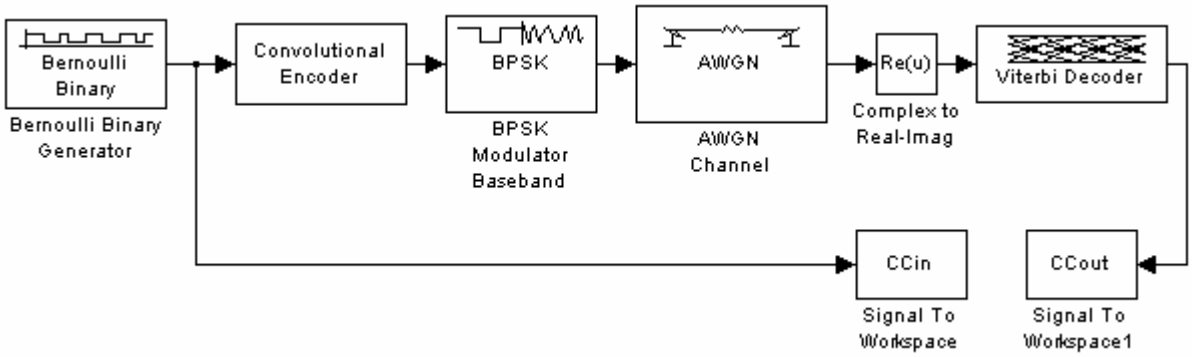
Pro správnou funkci takto upraveného modelu je nezbytné, aby na přijímací straně byly pozice odpovídající vynechaným bitům doplněny nulami. K tomuto účelu slouží blok Insert Zero, který do procházejícího datového toku vkládá nuly a to podle pravidel definovaných pomocí vektoru. Hodnota 0 v tomto vektoru udává pozici v datovém toku, na kterou bude nula vložena. V našem případě to tedy znamená zajistit, aby tento vektor byl totožný s děrovacím vektorem, čímž bude docíleno vkládání nul na pozice, jejichž hodnoty byly vyloučeny z dalšího přenosu v bloku děrování.

U bloků Puncture i Insert Zero platí, že pokud je procházející datový tok delší než zadaný děrovací vektor (to nastává ve většině případů), bude se tento vektor postupně opakovat na celý datový tok. Jedinou podmínkou zůstává, aby délka tohoto toku byla dělitelná beze zbytku délkou děrovacího vektoru.

4.7 Model zabezpečovacího procesu u konvolučních kódů

Model pro simulaci zabezpečovacího procesu u konvolučních kódů vychází z obrázku 4.11. Blok Bernoulli Binary Generátor generuje vstupní datovou posloupnost, která následně přichází na vstup konvolučního kodéru. Na zakódovaná data je poté použita BPSK modulace v základním pásmu a takto upravená posloupnost vstupuje do přenosového kanálu. Pro účely této simulace se opět využívá model AWGN přenosového kanálu. Na přijímací straně stačí z přijatého signálu odebírat reálnou složku (není potřeba přijatá data demodulovat), jelikož Viterbi dekodér je nastaven tak, aby při dekódovacím procesu využíval měkkého způsobu dekódování.

Tento model slouží pouze ke srovnání zabezpečovacích schopností konvolučních kódů a turbokódů použitých v následující kapitole.



Obr. 4.11: Model pro simulaci zabezpečovacího procesu u konvolučních kódů.

5 Prezentace výsledků simulace

Tato kapitola se věnuje prezentaci výsledků simulací popsaných v předchozí kapitole. Bude zde postupně rozebírán vliv jednotlivých parametrů, které zásadním způsobem ovlivňují výslednou korekční schopnost turbokódů. Při návrhu turbokódu je nutné nejvíce zohlednit následující parametry:

- Počet dekódovacích kroků.
- Použití bloku děrování.
- Generující polynomy a délku kódového ohraničení RSC kodérů.
- Velikost vstupního bloku dat.
- Použití vhodného bloku prokládání.
- Výběr vhodného dekódovacího algoritmu.

5.1 Vliv počtu dekódovacích kroků

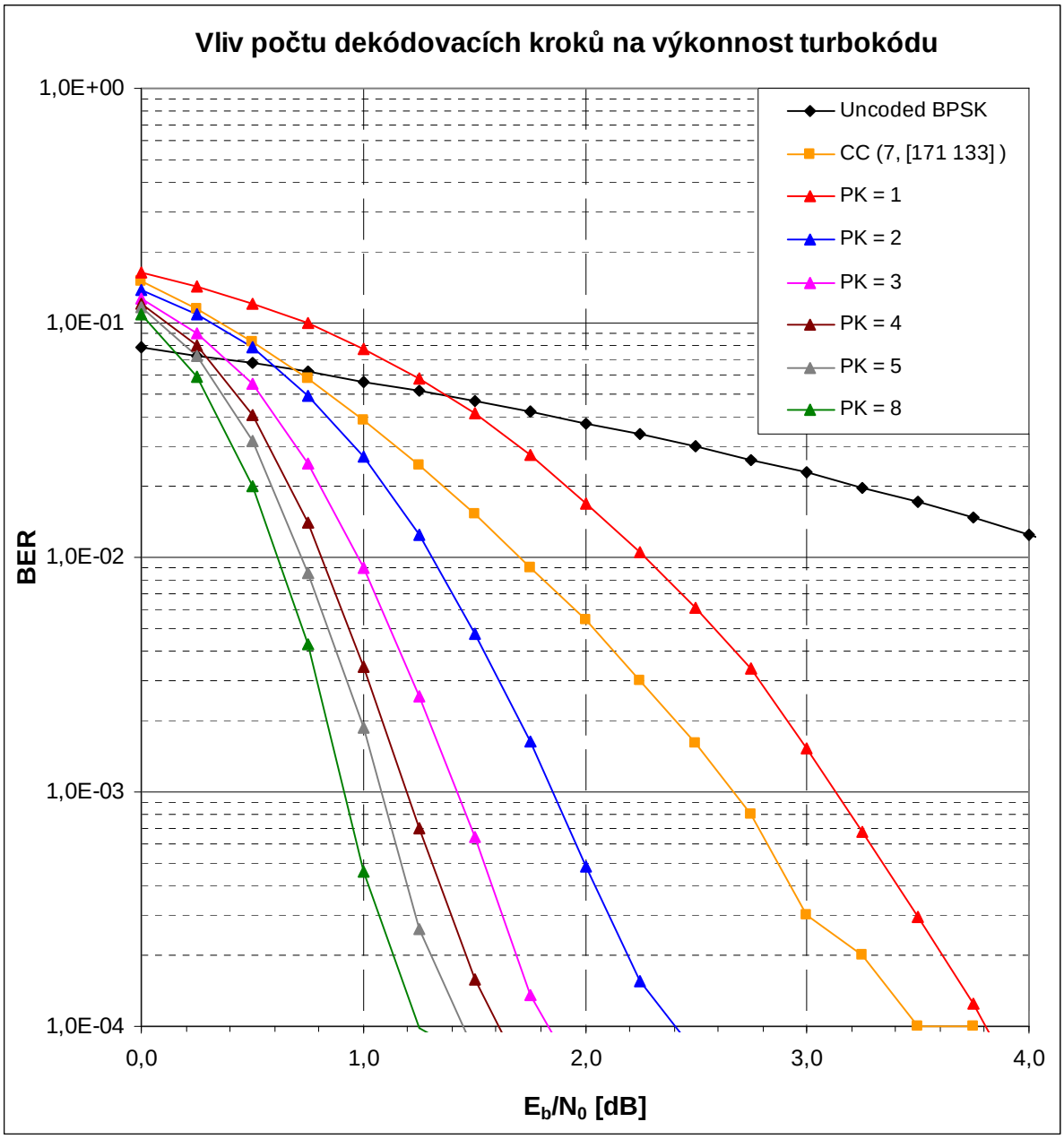
Tato simulace vychází ze základního modelu uvedeného na obrázku 4.1. Turbokodér se skládá ze dvou identických paralelně zřetězených RSC kodérů, vzájemně oddělených pseudonáhodným prokladačem. Dílčí RSC kodéry mají délku kódového ohraničení $K = 4$ a generující polynomy jsou $G_0 = 15$ (rekurzivní) a $G_1 = 17$ (hodnoty generujících polynomů se opět uvádějí v osmičkové soustavě). Délka vstupního bloku dat byla rovna $L = 1000$ bitů a celkový počet vstupních bitů je $N = 500000$. K realizaci iterativního způsobu dekódování bude v rámci celé této diplomové práce využíván SOVA algoritmus.

Obrázek 5.1 umožňuje porovnat výkonnost výše popsaného turbokódu v závislosti na zvyšujícím se počtu dekódovacích kroků. Pro možnost srovnání jsem zde uvedl i případ nekódovaného BPSK a také konvolučního kódu s délkou kódového ohraničení $K = 7$, jehož generující polynomy jsou $G_0 = 171$ a $G_1 = 133$. Z obrázku vyplývá, že při rostoucím počtu dekódovacích kroků se výrazným způsobem zvyšuje výkonnost (korekční schopnost) tohoto turbokódu.

Pro $BER = 10^{-4}$ je vidět zlepšení mezi jedním a dvěma dekódovacími kroky přibližně o 1,4 dB. Toto zlepšení mezi jednotlivými kroky se ovšem s jejich rostoucím počtem výrazně snižuje. Rozdíl mezi osmi a pěti kroky už činí pouze 0,2 dB. Následující tabulka obsahuje ukázku klesajícího počtu chyb při rostoucím počtu dekódovacích kroků. Na posledním řádku této tabulky je možné sledovat výrazné zlepšovaní korekčních schopností testovaného turbokódu v závislosti na rostoucím počtu kroků při dekódování. Například pro $PK = 1$ bylo zjištěno 20570 chyb, kdežto v případě $PK = 8$ pouhých 36 chyb z celkového počtu 500000 vstupních bitů.

E_b/N_0 [dB]	PK =1	PK =2	PK =3	PK =4	PK =5	PK =8
0,00	82126	69443	63990	60778	58730	54547
0,25	71443	54227	45329	39810	35910	29349
0,50	60718	39253	27606	20324	15739	10034
0,75	49848	24388	12641	6973	4320	2129
1,00	38955	13366	4507	1712	942	229
1,25	28997	6280	1281	348	130	49
1,50	20570	2375	323	79	39	36

Tab. 5.1: Počet chyb v jednotlivých dekódovacích krocích.



Obr. 5.1: Vliv počtu dekódovacích kroků.

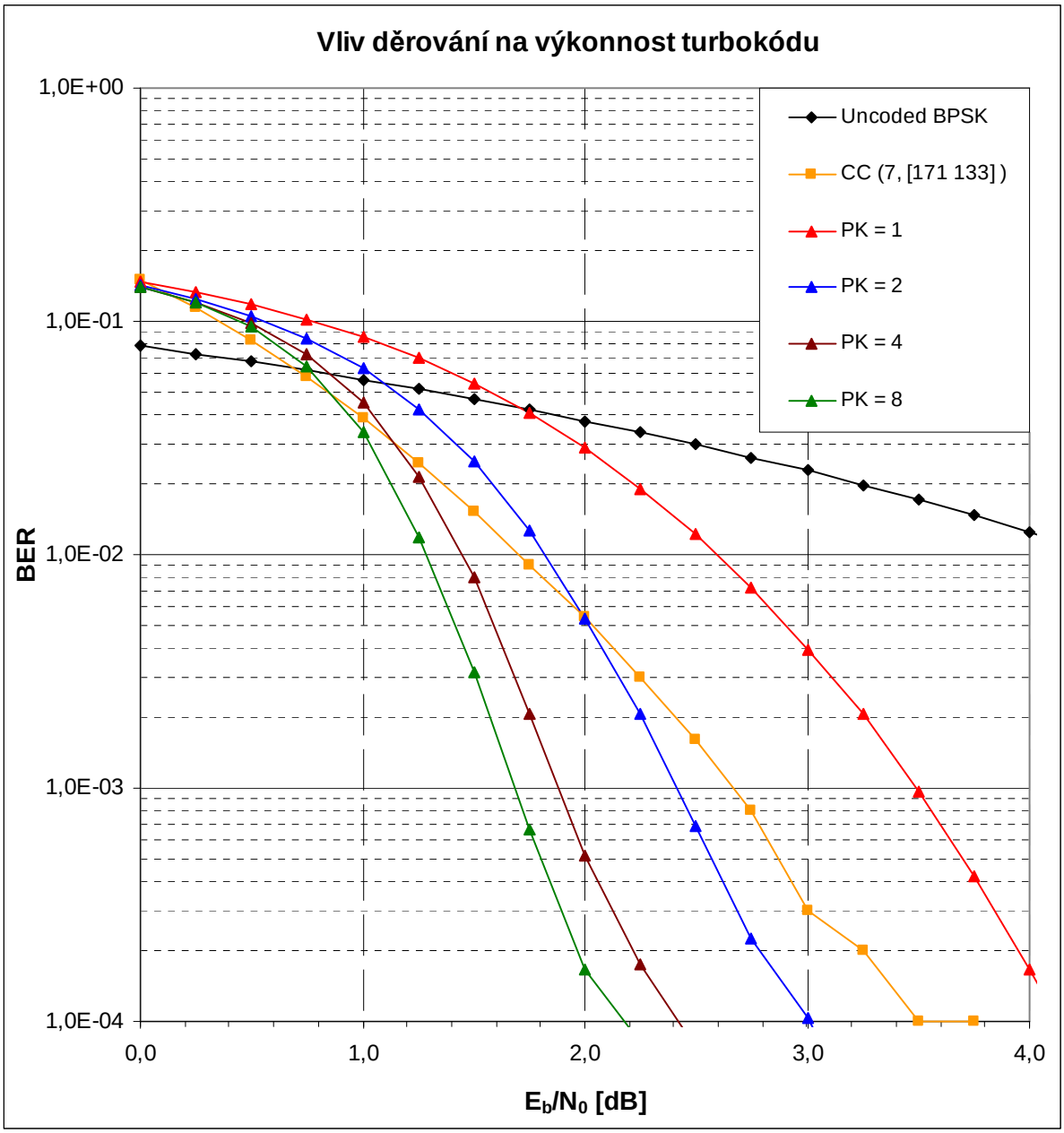
5.2 Vliv děrování na výkonnost turbokódu

V této simulaci se vychází z modelu uvedeného na obrázku 4.9, který rozšiřuje základní model z obrázku 4.1 o děrování paritních posloupností dílčích RSC kodérů. Byl zvolen děrovací vzor [110101] (systematický bit se přenáší vždy a ke každému systematickému bitu je přidán pouze jeden zabezpečovací bit náležející střídavě k prvnímu nebo ke druhému RSC kodéru). Dochází tedy k děrování zabezpečovacích posloupností jednotlivých RSC kodérů, kdy jsou dále přenášeny pouze liché bity paritní posloupnosti prvního a sudé bity paritní posloupnosti druhého RSC kodéru. Výsledkem je pokles celkové informační rychlosti turbokodéru z hodnoty $R = 1/3$ na $R = 1/2$. Ostatní parametry simulace zůstávají stejné, jako v předchozím případě. Ve všech prezentovaných výsledcích této kapitoly se předpokládá použití pseudonáhodného bloku prokládání. Výjimku tvoří část

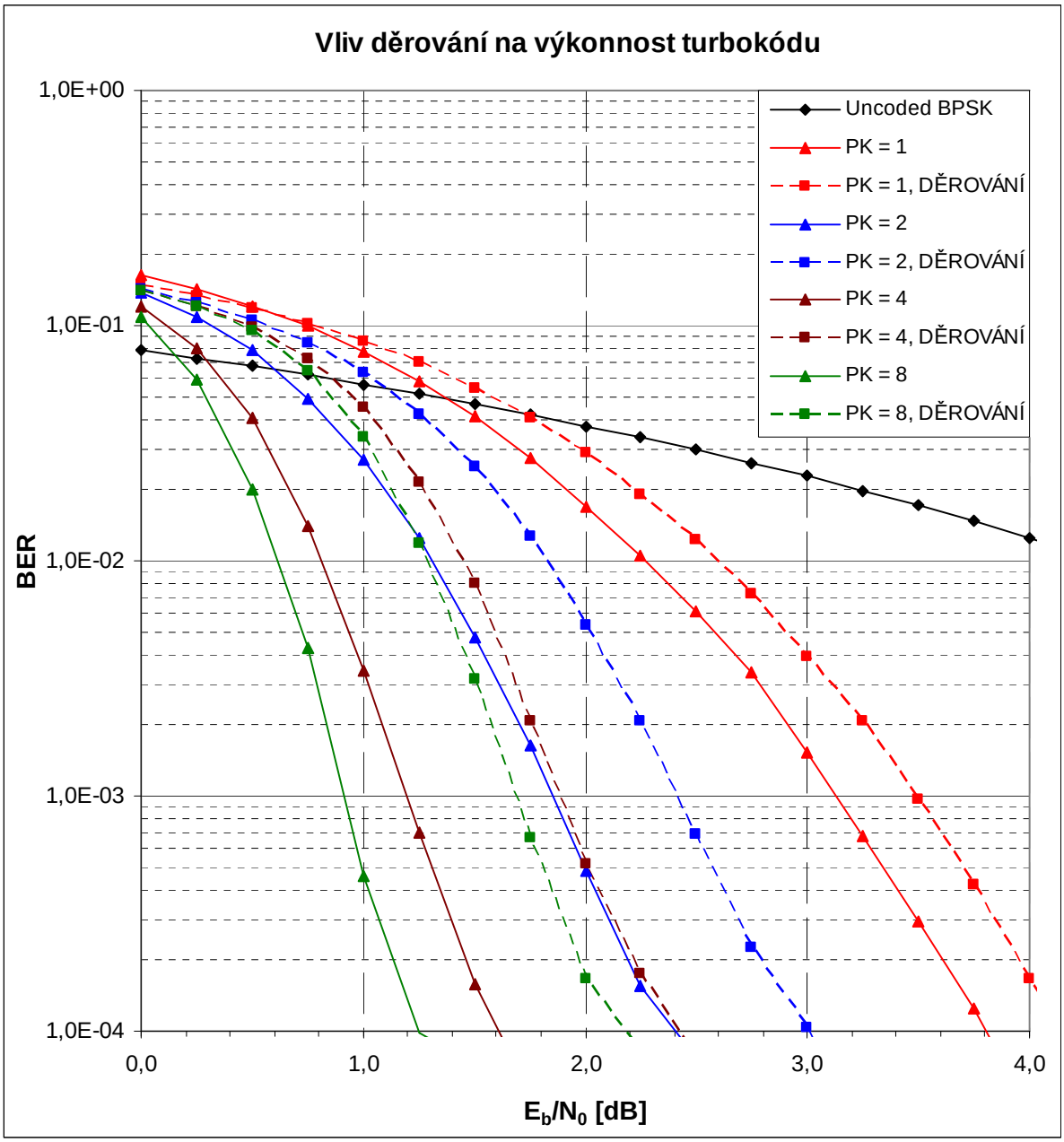
5.5, kde bude zkoumán vliv klasického blokového prokladače na zabezpečovací schopnost turbokódu.

Obrázek 5.2 ilustruje vliv rostoucího počtu dekódovacích kroků na výkonnost takto děrovaného turbokódu. Lze pozorovat výrazné zhoršení zabezpečovacích schopností, oproti výsledkům z obrázku 5.1, což je nutná cena za odpovídající zvýšení výsledné informační rychlosti turbokodéru.

Detailnější porovnání vlastností obou turbokódů nabízí obrázek 5.3. Z tohoto obrázku lze odvodit, že vzájemný rozdíl mezi výkonností obou turbokódů narůstá s počtem dekódovacích kroků. Pro $PK = 1$ činí rozdíl mezi děrovaným a neděrovaným turbokódem asi 0,4 dB, kdežto při $PK = 8$ narůstá tento rozdíl už téměř na celý 1 dB (odečítáno pro $BER = 10^{-4}$). Dále si můžeme všimnout, že při použití neděrovaného turbokódu s $PK = 2$ a děrovaného turbokódu s $PK = 4$ jde v našem případě dosáhnout srovnatelných výsledků.



Obr. 5.2: Vliv děrování na výkonnost turbokódů.



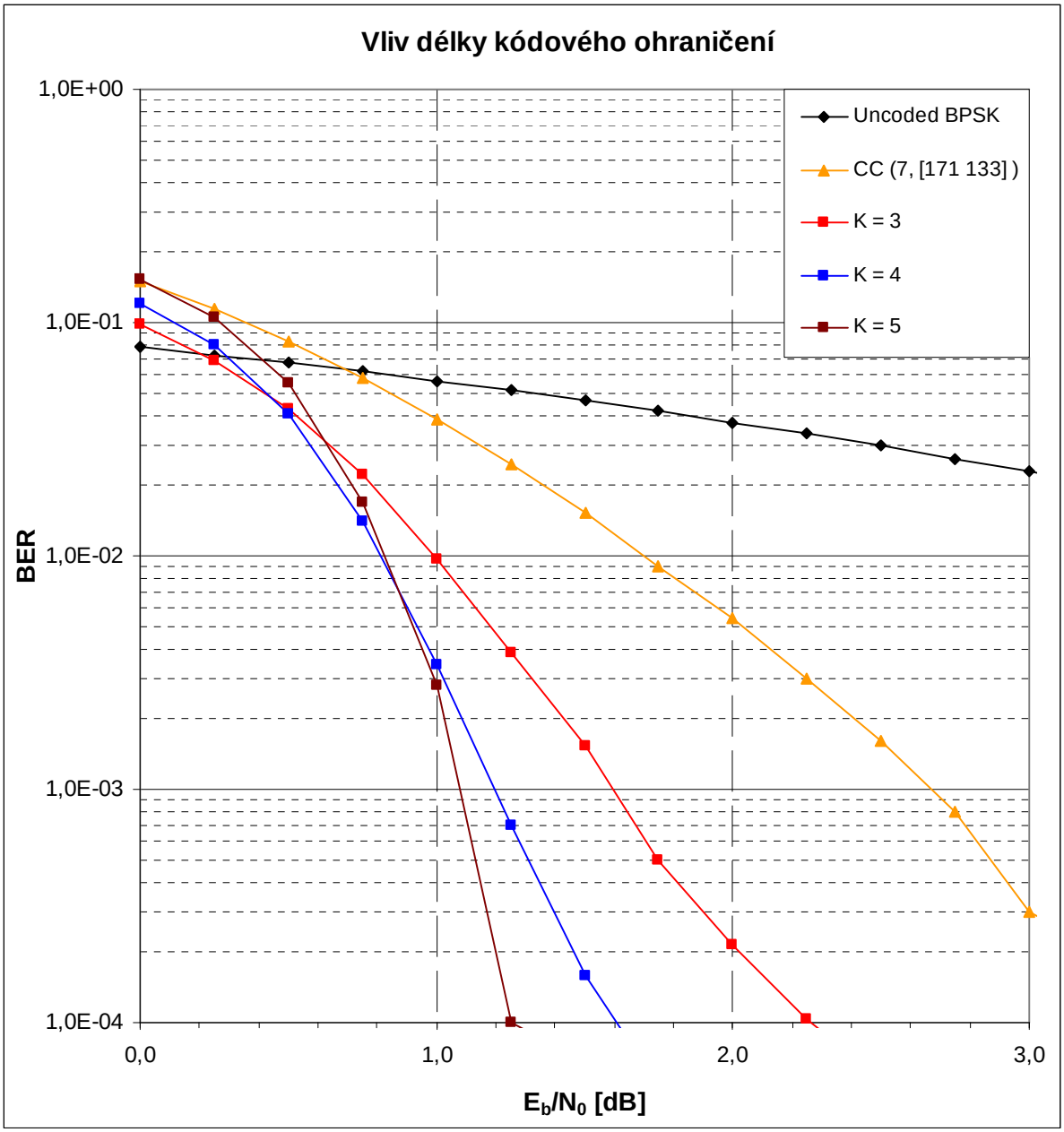
Obr. 5.3: Vliv děrování na výkonnost turbokódů - srovnání.

5.3 Vliv změny délky kódového ohraničení RSC kodérů

Tato simulace se zaměřuje na srovnání zabezpečovacích schopností turbokódů v případě, kdy budeme postupně měnit délku kódového ohraničení dílčích RSC kodérů použitých v bloku turbokodéru. Délka kódového ohraničení udává, jak dlouho se jednotlivý bit ze vstupní nezabezpečené posloupnosti podílí na zabezpečovacím procesu. Například pro RSC kodér s $K = 4$ a informační rychlost $R = 1/2$ bude každý vstupní bit takového kodéru ovlivňovat během zabezpečovacího procesu 4 páry výstupních zabezpečených bitů. Z teoretického hlediska lze tedy předpokládat, že rostoucí délka kódového ohraničení povede k nárůstu vzájemných vazeb (vzájemné provázanosti) mezi jednotlivými zakódovanými kombinacemi, což by mělo vést ke zvýšení celkové zabezpečovací schopnosti navrhovaného turbokódu. Tento předpoklad potvrzuje graf z obrázku 5.4, kde je patrný nárůst korekčních schopností v případě zvyšování délky kódového ohraničení u použitých RSC kodérů.

Simulace opět vychází ze základního modelu na obrázku 4.1 (tedy bez použití děrování). Při iterativním dekódování bylo pro všechny tři zobrazené průběhy použito čtyř dekódovacích kroků ($PK = 4$). Délka vstupního bloku dat, stejně jako i celkový počet vstupních bitů, je shodný s oběma předchozími případy.

Každá zvolená délka kódového ohraničení samozřejmě nabízí spoustu různých kombinací pro volbu generujících polynomů navrhovaného kodéru. Literatura [16] obsahuje přehled nejvýkonnějších konvolučních kódů s kódovou rychlostí $R = 1/2$ při dané délce kódového ohraničení. Poznatky z této literatury jsou uvedeny v tabulce 5.2. Symbol $\mathbf{d_{free}}$ (free distance) je v případě konvolučních kódů ekvivalentem minimální Hammingovy vzdálenosti ($\mathbf{d_{min}}$), známé z blokových kódů (minimální Hammingova vzdálenost kódu se definuje jako nejmenší Hammingova vzdálenost z množiny všech Hammingových vzdáleností mezi jednotlivými kódovými slovy).



Obr. 5.4: Vliv délky kódového ohraničení.

Parametr $\mathbf{d_{free}}$ u konvolučních kódu tedy udává nejmenší Hammingovu vzdálenost mezi různými dvojicemi kódových sekvencí, které ovšem musí začínat a končit v jednom stavu. Jelikož konvoluční kódy patří mezi kódy lineární, stačí brát v úvahu pouze ty sekvence, jejichž začátek i konec je v nulovém stavu (všechny paměťové buňky sledovaného kodéru obsahují nuly). Symbol **G** v tabulce 5.2 značí kódový zisk (označovaný jako asymptotic coding gain), který udává maximální hodnotu rozdílu výkonnosti mezi nekódovaným BPSK a použitým konvoluční kódem. Pro jeho výpočet platí vztah:

$$G = 10.\log_{10}\left(d_{free}.R_c\right). \tag{5.2.1}$$

Graf na obrázku 5.4 ukazuje vliv rostoucí délky kódového ohraničení RSC kodérů na zabezpečovací schopnost navrhovaného turbokódu. Generující polynomy pro danou hodnotu K byly zvoleny podle tabulky 5.2. Z uvedeného grafu vyplývá, že rozdíl mezi výkonností turbokódů s $K = 3$ a $K = 4$ činí zhruba 0,65 dB, kdežto mezi turbokódy s $K = 4$ a $K = 5$ je rozdíl 0,38 dB (odečítáno opět pro $BER = 10^{-4}$). Další nárůst délky kódového ohraničení už nepovede k výraznému zlepšování korekčních schopností turbokódu, naopak se bude prudce zvyšovat výpočetní náročnost dekódovacího procesu.

K	Polynomy	d_{free}	G [dB]
3	(7, 5)	5	3,98
4	(15, 17)	6	4,77
5	(23, 35)	7	5,44
6	(53, 75)	8	6,02
7	(133, 171)	10	6,99
8	(247, 371)	10	6,99

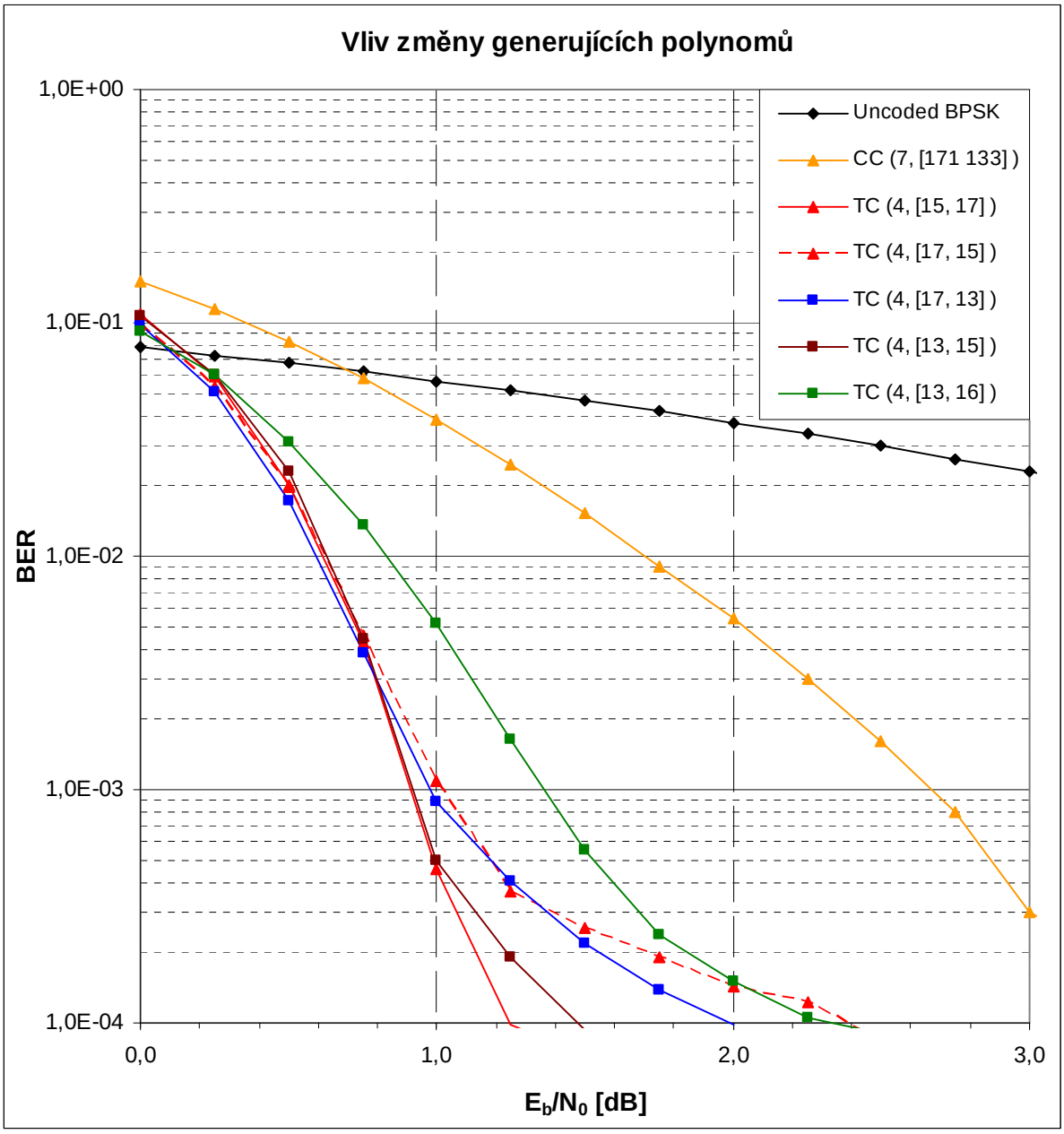
Tab. 5.2: Nejvýkonnější konvoluční kódy s kódovou rychlostí $R = 1/2$.

5.4 Vliv správné volby generujících polynomů

Tabulka 5.2 obsahuje generující polynomy nejvýkonnějších konvolučních kódů s kódovou rychlostí $R = 1/2$. Následující graf zobrazuje vliv změny těchto generujících polynomů na zabezpečovací schopnost turbokódu. Při simulaci jsou použity RSC kodéry s délkou kódového ohraničení $K = 4$. Ostatní parametry této simulace jsou: $L = 1000$, $N = 500000$, $PK = 8$ a není použito děrování.

Jako první umožňuje obrázek 5.5 sledovat rozdíl mezi turbokódy s generujícími polynomy (15, 17) a (17, 15). U klasických (NRC) konvolučních kódů nemá vzájemná výměna těchto polynomů žádný vliv na jeho celkovou výkonnost, kdežto v případě turbokódů způsobí záměna jednotlivých generujících polynomů pokles výkonnosti přibližně o 1,2 dB. Tento výrazný pokles způsobuje právě použití RSC kodérů jako základních stavebních prvků turbokodéru (v rámci celé této kapitoly odpovídá první udávaná hodnota zpětnovazebnímu generujícímu polynomu). Z tohoto důvodu je při návrhu turbokódu nutné zohlednit i pořadí vytvářecích mnohočlenů.

Dalším cílem této simulace bylo ověřit platnost údajů uvedených v tabulce 5.2, tedy skutečnost, že volbou generujících polynomů (15, 17) v RSC kodérech bude dosaženo nejlepších výsledků pro délku kódového ohraničení $K = 4$. K tomuto účelu jsem náhodně zvolil další dvojice polynomů (17, 13), (13, 15) a (13, 16). Srovnání dílčích výsledků simulace ukazuje graf 5.5, ze kterého je patrné, že volba polynomů (15, 17) vede opravdu k zajištění nejlepších zabezpečovacích schopností turbokódu pro $K = 4$.

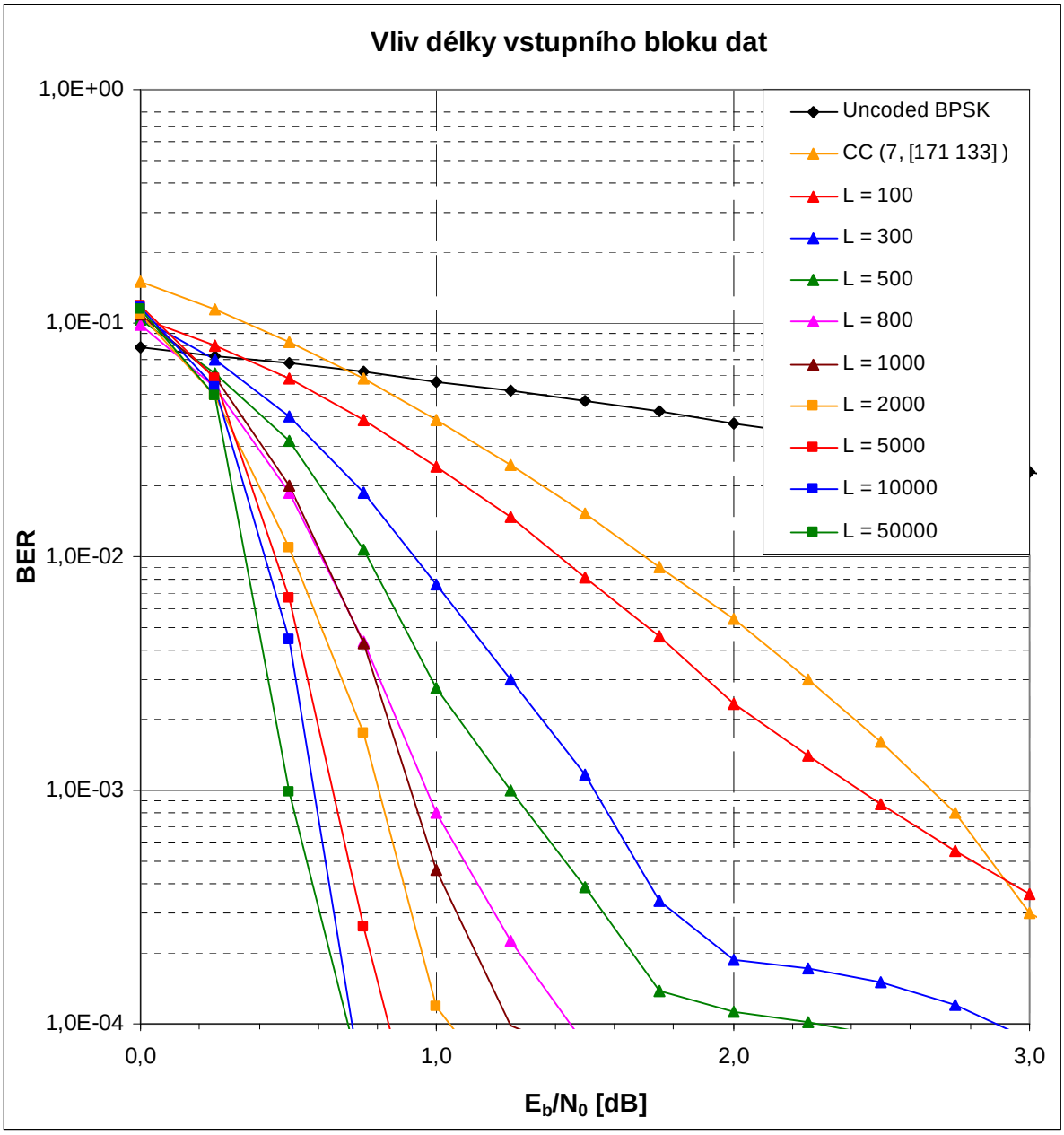


Obr. 5.5: Vliv volby generujících polynomů.

5.5 Vliv velikosti vstupního bloku dat

Velikost vstupního bloku je jedním z parametrů, které nejvíce ovlivňují celkovou výkonnost navrhovaného turbokódu. Graf 5.6 ukazuje nárůst korekčních schopností turbokódu v závislosti na rostoucí délce vstupního bloku dat. Pro tuto simulaci byly zvoleny RSC kodéry s generujícími polynomy (15, 17) a délkou kódového ohraničení $K = 4$. Ostatní parametry jsou: $N = 500000$, $PK = 8$ a celková informační rychlost turbokódu je $R_c = 1/3$ (není požito děrování).

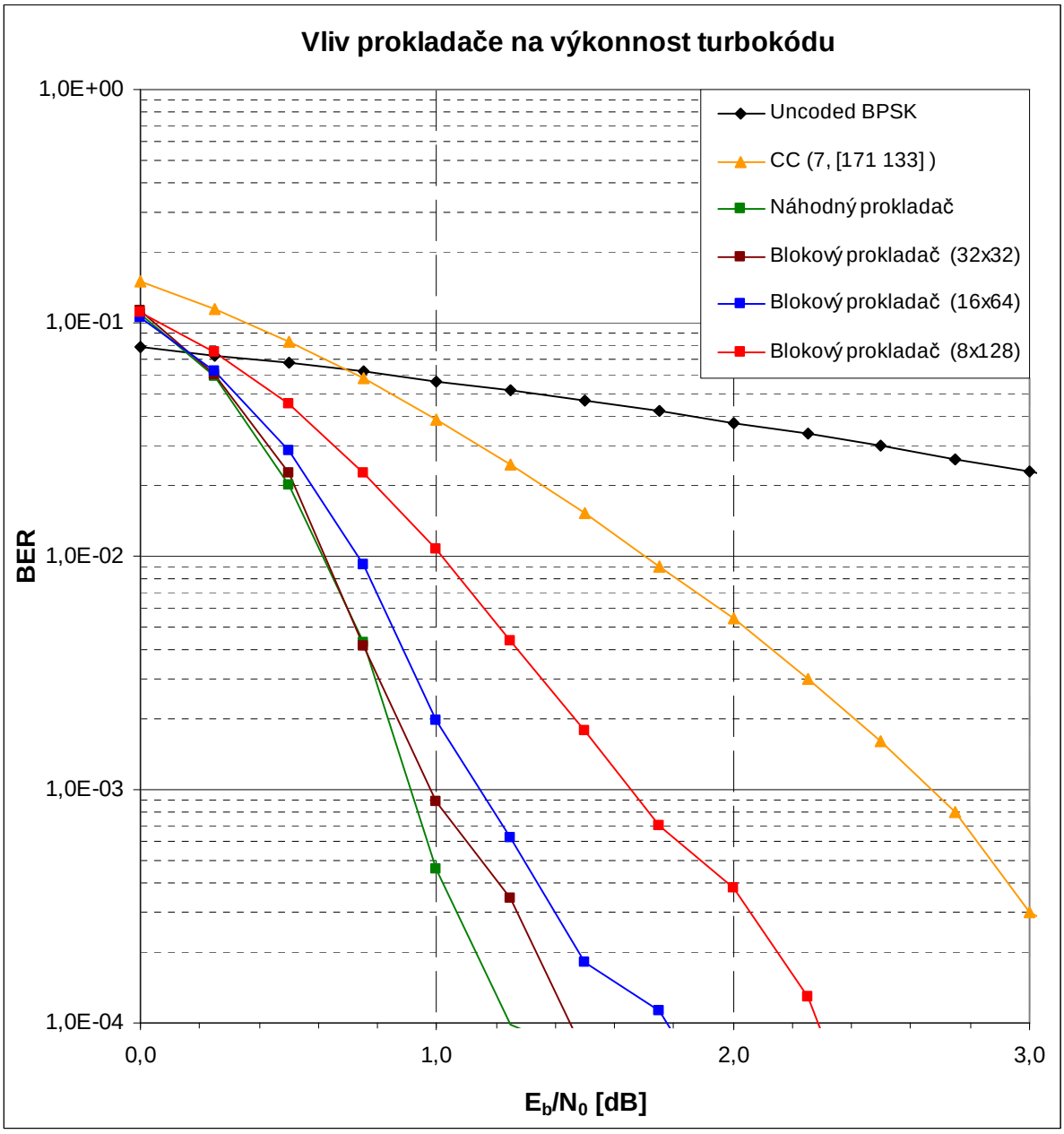
Graf 5.6 potvrzuje teoretický předpoklad, že s rostoucí délkou vstupního bloku dat bude docházet k výraznému zlepšování korekčních schopností navrhovaného turbokódu. S rostoucí délkou vstupního bloku ovšem úměrně narůstá i celkové zpoždění při dekódování jednotlivých bloků, proto je nutné při návrhu turbokódu hledat kompromis mezi výkonností daného turbokódu a akceptovatelným zpožděním.



Obr. 5.6: Vliv délky vstupního bloku dat.

5.6 Vliv použitého prokladače na výkonnost turbokódu

Význam prokládání a jeho vliv na celkovou zabezpečovací schopnost daného turbokódu byl podrobněji popsán v kapitole 1.7. V této části bylo taktéž uvedeno, že nejlepší výkonnosti dosahují turbokódy při použití náhodného nebo pseudonáhodného prokladače. Následující simulace slouží k ověření tohoto teoretického předpokladu. Graf 5.7 srovnává zabezpečovací schopnosti turbokódu s náhodným prokladačem a turbokódu, který využívá klasický blokový prokladač. V případě náhodného prokladače byla použita délka vstupního bloku dat $L = 1000$ bitů a celkový počet vstupních bitů je $N = 500000$, kdežto pro všechny tři varianty blokového prokladače bylo $L = 1024$ a $N = 499712$ bitů. Pro simulaci jsem zvolil rozměry blokového prokladače (32x32), (16x64), (8x128), přičemž všechny tyto simulované charakteristiky jsou měřeny při $PK = 8$. Délka kódového ohraničení a generující polynomy RSC kodéru zůstávají stejné, jako v předchozím případě.



Obr. 5.7: Vliv prokladače na výkonnost turbokódu.

Graf 5.7 ukazuje zlepšení výkonnosti o 0,2 dB mezi turbokódem s náhodným blokem prokládání a turbokódem s blokovým prokladačem (32x32). Dále je možné sledovat výrazný pokles zabezpečovacích schopností turbokódu v závislosti na klesající hloubce prokládání u blokového prokladače. Rozdíl mezi použitím blokového prokladače s rozměrem (32x32) a (8x128) dosahuje hodnoty zhruba 1 dB při $BER = 10^{-4}$.

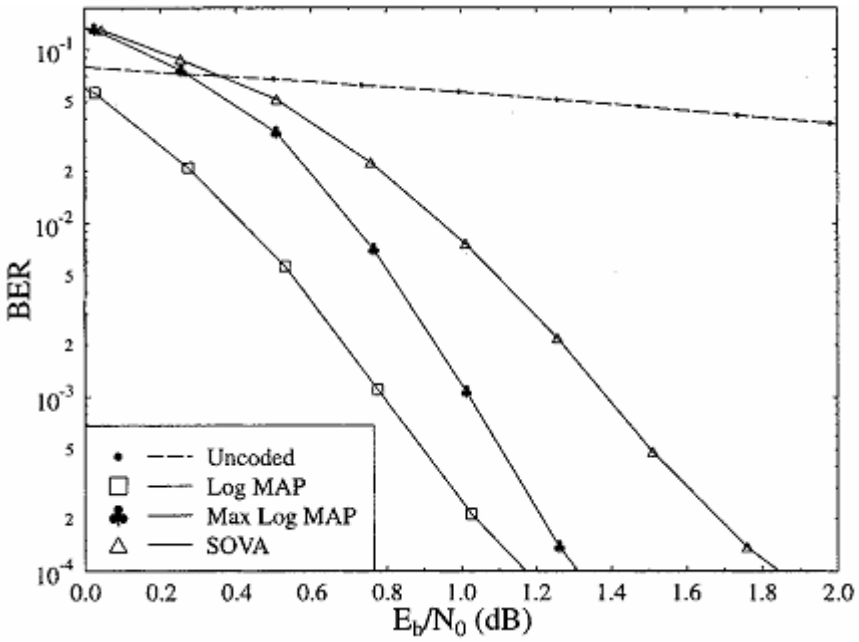
5.7 Vliv použitého dekódovacího algoritmu

Všechny dříve uvedené simulace používaly k iterativnímu dekódování turbokódů SOVA algoritmus, který byl podrobně rozebírán v kapitole 2.4. Druhá kapitola této práce se věnovala problematice dekódování turbokódů a kromě SOVA algoritmu byl jako další možná alternativa představen i MAP algoritmus. MAP algoritmus není v turbodekodérech často používán, a to díky jeho vysoké výpočetní náročnosti. Častěji jsou implementovány

modifikované verze tohoto algoritmu označované jako Log-MAP a Max-Log-MAP, jelikož výrazným způsobem redukují výpočetní náročnost původního MAP algoritmu (více informací lze najít v literatuře [9] a [10]).

Dá se předpokládat, že volba vhodného algoritmu pro iterativní dekódování turbokódů bude významným způsobem ovlivňovat jejich celkovou zabezpečovací schopnost. Naprogramování všech těchto dekódovacích algoritmů a následné otestování jejich vlivu na předcházející simulace přesahuje rámec této diplomové práce, přesto uvádím alespoň jednu ukázku závislosti korekčních schopností turbokódu na zvoleném typu dekódovacího algoritmu. Tento graf byl převzat z literatury [17], která se věnuje dané problematice. Parametry k této simulaci byly (podle autorů tohoto článku) následující: délka vstupního bloku dat $L = 1000$ bitů, celková informační rychlost testovaného turbokódu $R_c = 1/3$, počet dekódovacích kroků $PK = 8$, délka kódového ohraničení RSC kodérů $K = 3$ s generujícími polynomy (7, 5). K prokládání vstupních dat je použit náhodný prokladač a celkový počet vstupních bloků byl 40000 (tedy 40 milionů vstupních bitů).

Grafu 5.8 ukazuje, že rozdíl mezi výkonností testovaného turbokódu při použití dekódovacích algoritmů SOVA a Max-Log-MAP činí přibližně 0,55 dB. V případě SOVA a Log-MAP algoritmů je tento rozdíl dokonce 0,7 dB (odečítáno pro hodnotu $BER = 10^{-4}$). Využití jiného algoritmu pro iterativní dekódování ve všech dříve uvedených simulacích by tedy vedlo ke zlepšení prezentovaných výsledků průměrně o dalších 0,6 dB.

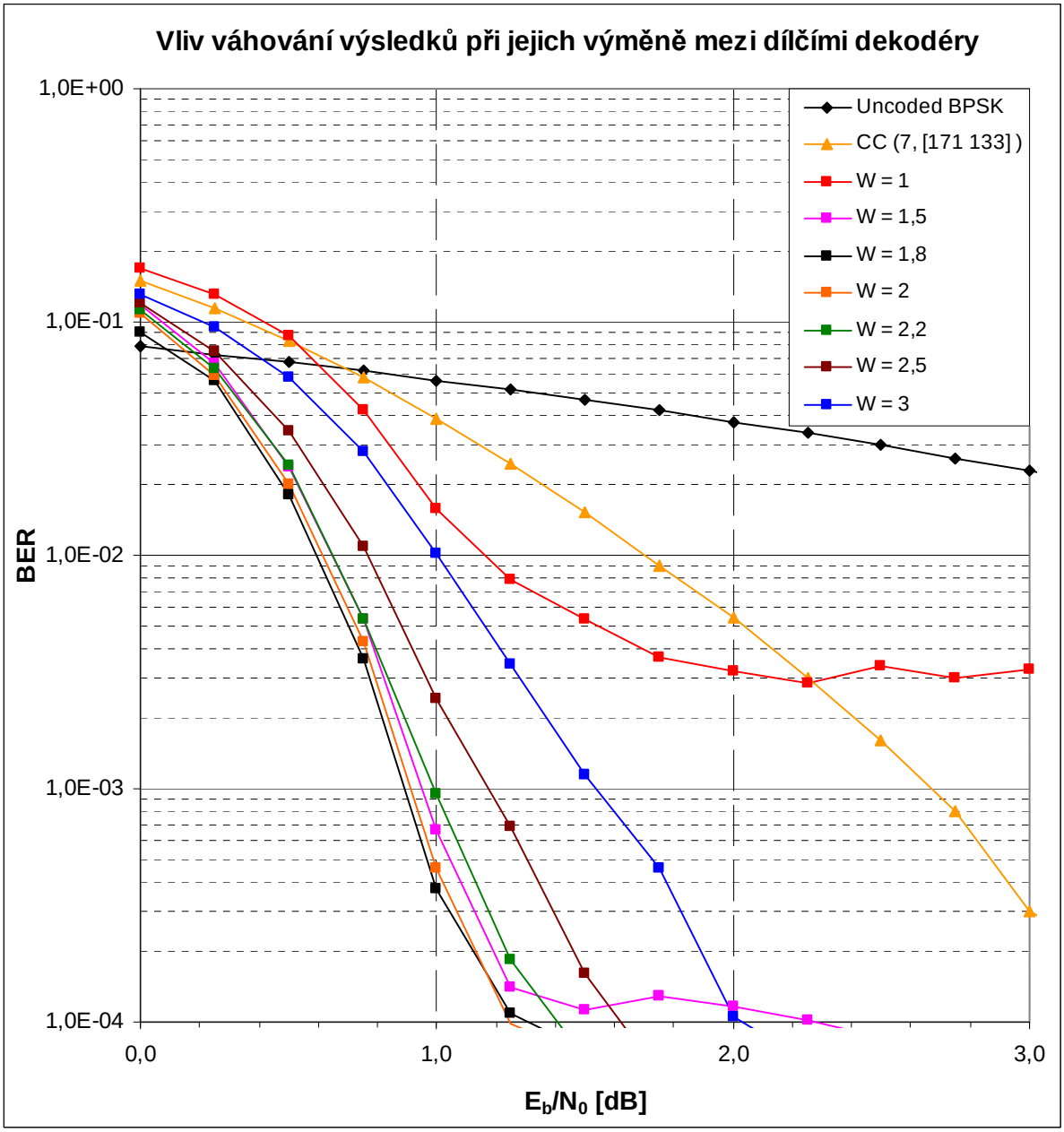


Obr. 5.8: Vliv použitého dekódovacího algoritmu – převzato z literatury [17].

5.8 Popis parametru W a jeho vliv na výkonnost turbokódu

Na závěr této kapitoly ještě uvádím důvod zavedení parametru W jako váhového koeficientu, určeného pro dělení apriorních informací při jejich vzájemné výměně mezi dílčími dekodéry v rámci iterativního způsobu dekódování. Dostupné literární prameny, které se věnují problematice iterativního dekódování turbokódů [4, 10 a 17], vycházejí při odvozování SOVA algoritmu z matematického popisu uvedeného v kapitole 2.4. Stejným způsobem jsem postupoval i při tvorbě SOVA algoritmu pro účely této diplomové práce, jehož správná funkce byla ověřena jednak na základě ručně spočítaného příkladu z kapitoly 2.4 a jednak z ukázkového příkladu v literatuře [17].

Problémy nastaly v okamžiku, kdy bylo potřeba dekódovat větší počet vstupních bloků. Následující obrázek demonstruje vliv dělení získaných apriorních informací váhovým koeficientem W na korekční schopnost turbokódu. Parametry simulace byly v tomto případě shodné s předchozími simulacemi ($L = 1000$, $N = 500$, $PK = 8$, $R_c = 1/3$, RSC kodéry s $K = 4$ a polynomy (15, 17)). Při volbě $W = 1$ (tedy bez použití váhového koeficientu) lze vidět celkovou degradaci zabezpečovacích schopností testovaného turbokódu. Postupnou změnou tohoto parametru a zkoumáním příslušné odezvy v chování turbokódu jsem se snažil nalézt takovou hodnotu W, která by byla schopna poskytnout stabilní výsledky i pro měnící se počet dekódovacích kroků. Graf 5.9 ukazuje, že nejlepších výsledků je dosaženo pro hodnotu $W = 2$ (všechny simulace z této kapitoly měly nastaven parametr W na hodnotu 2). Nepodařilo se mi ovšem najít odbornou publikaci, která by se věnovala tomuto problému.



Obr. 5.9: Vliv parametru W na výkonnost turbokódu.

6 Závěr

Tato diplomová si klade za cíl podat ucelený přehled o problematice zabezpečení přenášených dat pomocí turbokódů. První kapitoly se detailně zaměřují na proces kódování a dekódování turbokódů. Jsou zde postupně popsány dílčí části turbokodéru i turbodekodéru včetně jejich vzájemných vazeb na výslednou zabezpečovací schopnost navrhovaného turbokódu. Po tomto teoretickém úvodu následuje kapitola zaměřená na využití těchto korekčních kódů v současných sdělovacích systémech. V této kapitole jsem se podrobněji zabýval turbokódy, které jsou aplikovány do nejznámějších sdělovacích systémů, jako například UMTS, cdma2000, DVB-RCS, DVB-RCT a CCSDS.

Zbývající část této práce je věnována ověření získaných teoretických předpokladů ohledně chování a zabezpečovacích schopností těchto kódů. Čtvrtá kapitola obsahuje seznam modelů (vytvořených v programu SIMULINK) včetně popisu a nastavení jednotlivých bloků. Dále jsou zde popsány i některé stěžejní části obslužného skriptu v programu MATLAB, jako například inicializace proměnných, kde je možné nastavovat a měnit parametry dané simulace, nebo ukázka části skriptu pro realizaci iterativního způsobu dekódování. Implementace SOVA algoritmu použitého v rámci této diplomové práce, vychází z algoritmu, jehož zdrojový kód je volně ke stažení na oficiálních webových stánkách k programu MATLAB [15]. Tento algoritmus byl upraven do podoby, která odpovídá popisu SOVA algoritmu ve druhé kapitole této práce. Zdrojový kód funkce **sova**, stejně jako všechny uvedené modely a obslužné skripty, je možné vyhledat na přiloženém CD.

Poslední kapitola zkoumá vliv různých parametrů navrhovaného turbokódu na jeho výsledné korekční schopnosti. Postupně byly testovány následující vlastnosti:

- Výkonnost turbokódu při měnícím se počtu dekódovacích kroků.
- Výkonnost turbokódu s děrováním paritních posloupností v turbokodéru ($R_c = 1/2$) a srovnání získaných výsledků s turbokódem bez děrování ($R_c = 1/3$).
- Výkonnost turbokódu v závislosti na rostoucí délce kódového ohraničení RSC kodérů.
- Výkonnost turbokódu při změně generujících polynomů RSC kodérů.
- Výkonnost turbokódu pro různé délky vstupního bloku dat.
- Výkonnost turbokódu v závislosti na použitém způsobu prokládání.
- Výkonnost turbokódu při použití rozdílných algoritmů pro iterativní dekódování.

Výsledné grafy jsou prezentovány v předchozí kapitole, kde je k nim přidán i popis nastavení jednotlivých parametrů, které byly použity při dané simulaci. Ze získaných charakteristik lze vyvodit následující obecné závěry:

- Pro pevně daný turbokodér se korekční schopnosti navrhovaného turbokódu zlepšují:
  - S narůstající délkou vstupního bloku dat.
  - S rostoucím počtem kroků při iterativním dekódování.
- Pro pevně stanovenou délku vstupního bloku dat a počet dekódovacích kroků je možné dále vylepšit zabezpečovací schopnosti daného turbokódu:
  - Zvětšováním délky kódového ohraničení RSC kodérů.
  - Volbou vhodnějších generujících polynomů RSC kodérů.
- V případě použití bloku děrování dochází vždy k poklesu korekčních schopností turbokódu, což je cena za odpovídající snížení výsledné přenosové rychlosti a tím i potřebné šířky pásma přenosového kanálu.
- Vhodný návrh prokladače vede ke zvýšení výkonnosti turbokódu, přičemž nejlepších výsledků je dosaženo při aplikaci náhodných nebo pseudonáhodných prokladačů.

- Správná volba dekódovacího algoritmu může výrazným způsobem zvýšit výslednou korekční schopnost navrhovaného turbokódu. Podle grafu 5.8 dosahuje největší výkonnosti Log-MAP algoritmus.

Z výše uvedených závěrů vyplývá, že je sice možné navrhnout opravdu robustní turbokód, který by se významně blížil teoretickému Shannonovu limitu pro daný přenosový kanál, ovšem rostoucí výkonnost turbokódu je vykoupena neúměrně velkým nárůstem zpoždění potřebného k dekódování. Například neustálým zvyšováním počtu dekódovacích kroků dochází při vyšších hodnotách (zhruba od 10 dekódovacích kroků) už pouze k nepatrnému zlepšení zabezpečovacích schopností turbokódu za cenu znatelného zvýšení zpoždění. Podobně je tomu v případě změny délky kódového ohraničení RSC kodérů, kdy rozdíl mezi $K = 3$ a $K = 4$ způsobí přibližně zdvojnásobení dekódovací náročnosti takového turbokódu a rozdíl mezi $K = 3$ a $K = 5$ povede už zhruba ke čtyřnásobnému zpoždění. Při návrhu vhodného turbokódu je tedy nezbytně nutné předem zvážit, v jaké oblasti bude tento turbokód aplikován (nároky na zpoždění, maximální povolená chybovost, atd.) a poté hledat kompromis mezi zabezpečovacími schopnostmi a odpovídajícím zpožděním.

Velkou výhodou zvolení programu MATLAB/SIMULINK pro simulování vlastností turbokódů bylo využití funkcí a jednotlivých bloků z Communication Toolboxu, který je součástí tohoto simulačního programu. To umožnilo především zjednodušení realizace kódovacího procesu turbokódů. V případě dekódovacího procesu, díky velké výpočetní náročnosti dekódovacího algoritmu, by bylo pravděpodobně vhodnější zvolit například programovací jazyk C, jelikož složitější simulace (např. velký počet dekódovacích kroků, dlouhé vstupní bloky dat, atd.) byly časově velmi náročné. Časová náročnost byla důvodem pro zvolení poměrně malého počtu vstupních bitů u prezentovaných simulací ($N = 500000$), stejně jako pro omezení sledovaného rozsahu BER pouze do hodnoty $BER = 10^{-4}$.

Komentář k zadání diplomové práce

Prvním cílem, kterého mělo být v této práci dosaženo, bylo vytvoření seznamu nejčastěji používaných turbokódů v současných sdělovacích systémech. Tento seznam je uveden na začátku třetí kapitoly, přičemž celá tato kapitola se podrobně věnuje nejčastěji využívaným turbokódům.

Druhým úkolem bylo prostudovat korekční schopnosti turbokódů a vlastnosti, které tuto schopnost nejvíce ovlivňují. Touto problematikou se zabývá pátá kapitola, kde je při jednotlivých simulacích zkoumán vliv změny různých parametrů na celkovou zabezpečovací schopnost navrhovaného turbokódu.

Ve třetí části zadání měl být vybrán vhodný turbokód (včetně některého z algoritmů pro iterativní dekódování) a měla být provedena simulace kódovacího a dekódovacího procesu. Postup při tvorbě simulace kódovacího a dekódovacího procesu byl detailně popsán ve čtvrté kapitole a výsledky těchto simulací jsou i s komentářem uvedeny v kapitole páté.

Posledním cílem bylo zaměřit se na možnost identifikace velikosti zabezpečovacích schopností turbokódu vzhledem k měnící se distribuci chyb v přenosovém kanálem. Na začátku přenosu se nejprve zjistí šumové poměry v přenosovém kanálu a na jejich základě se zvolí počet kroků pro iterativní dekódování tak, aby bylo dosaženo odpovídajících zabezpečovacích požadavků. Tabulka 5.1 z páté kapitoly ukazuje, jak výrazným způsobem se snižuje počet chyb při rostoucím počtu použitých dekódovacích kroků. Podobně i graf 5.1 ukazuje závislost BER na E_b/N_0 při rostoucím počtu dekódovacích kroků.

Tímto jsem přesvědčen o tom, že bylo splněno zadání diplomové práce, a to ve všech jeho bodech.

7 Použitá literatura

[1] BIOLEK, D., *Datová komunikace: Úvod do teorie informace a kódování*, Skripta VUT, 2002.

[2] HUANG, F., Chapter 2, *Convolutional Codes*, Dostupné z URL: *http://scholar.lib.vt.edu/theses/available/etd-71897-15815/unrestricted/chap2.pdf*.

[3] HUANG, F., Chapter 3, *Turbo Code Encoder*, Dostupné z URL: *http://scholar.lib.vt.edu/theses/available/etd-71897-15815/unrestricted/chap3.pdf*.

[4] HUANG, F., Chapter 4, *Iterative Turbo Decoder*, Dostupné z URL: *http://scholar.lib.vt.edu/theses/available/etd-71897-15815/unrestricted/chap4.pdf*.

[5] Učební texty, *Konvoluční kódování a dekódování*, Dostupné z URL: http://vyuka.fel.zcu.cz/kae/+pi/kody/Konvolucni%20kodovani.pdf.

[6] BERROU, C., GLAVIEUX, A., THITIMAJSHIMA, P., *Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1*, in Proc. IEEE International Conference on Communications (ICC '93), vol. 2, pp. 1064-1070, Geneva, Switzerland, May 1993.

[7] BERROU C., *The ten-year-old turbo codes are entering into service*, IEEE Commun. Mag., vol. 41, no. 8, pp. 110-116, 2003.

[8] SKLAR B., *Fundamentals of Turbo Codes*, Dostupné z URL: http://www.informit.com/content/images/art_sklar3_turbocodes/elementLinks/art_sklar3_turbocodes.pdf.

[9] SKLAR B., *Maximum a Posteriori Decoding of Turbo Codes*, Dostupné z URL: http://www.informit.com/content/images/art_sklar4_map/elementLinks/art_sklar4_map.pdf .

[10] DENNET, P., *An investigation of Turbo Codes over Mobile Wireless Channels*, University of Wolverhampton, Oct-2006. 225 s. Dizertační práce.

[11] WHITE, G., *Optimised Turbo Codes for Wireless Channels*, University of York, UK, Oct -2001. 217 s. Dizertační práce.

[12] DVB-RCS Standard, *Interaction channel for satellite distribution systems*, ETSI EN 301 790, V1.2.2, pp. 21-24, December 2000.

[13] MATTHEW, C., JIAN, S., Handbook of RF and Wireless Technologies, *Turbo codes*, Chapter 12, 375–399. F. Dowla Editor, Newnes Press, 2004.

[14] Communication Toolbox – Documentation, Dostupné z URL: http://www.mathworks.com/access/helpdesk/help/toolbox/comm/comm.shtml

[15] BOHDANOWIC, A., *Soft Input Soft Output Viterbi Algorithm*, Dostupné z URL: http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=3801&objectType=FILE.

[16] LE GOFF, S., *Information Theory & Coding – EEE2004*, School of EECE, Spring 2007.

[17] WOODARD, J., HANZO, L., *Comparative Study of Turbo Decoding Techniques: An Overview*, IEEE transactions on vehicular technology, vol. 49, no. 6, November 2000.